University of Alberta

Optimization Approaches For Large Scale Model Predictive Control

by

Folasade Bunmi Adebiyi

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of **Master of Science**.

in

Process Control

Department of Chemical and Materials Engineering

Edmonton, Alberta

Fall 2003

National Library of Canada
Acquisitions and Bibliographic Services
395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque nationale du Canada
Acquisisitons et services bibliographiques
395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence
ISBN: 0-612-87703-5
Our file Notre référence
ISBN: 0-612-87703-5

The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou aturement reproduits sans son autorisation.

Canada

University of Alberta

Library Release Form

Name of Author: Folasade Bunmi Adebiyi

Title of Thesis: Optimization Approaches For Large Scale Model Predictive Control

Degree: Master of Science

Year this Degree Granted: 2003

Permission is hereby granted to the University of Alberta Library to reproduce single copies of this thesis and to lend or sell such copies for private, scholarly or scientific research purposes only.

The author reserves all other publication and other rights in association with the copyright in the thesis, and except as hereinbefore provided, neither the thesis nor any substantial portion thereof may be printed or otherwise reproduced in any material form whatever without the author's prior written permission.

Date: August 5, 2003

University of Alberta

Faculty of Graduate Studies and Research

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research for acceptance, a thesis entitled **Optimization Approaches For Large Scale Model Predictive Control** submitted by Folasade Bunmi Adebiyi in partial fulfillment of the requirements for the degree of **Master of Science** in *Process Control.*

Dr. Ming J. Zuo

Date: August 8, 2003

Abstract

This work deals with the analysis of optimization algorithms for large scale Model Predictive Control(MPC). Different solution strategies are developed, analyzed and implemented on MPC case studies.

Model Predictive control has rapidly found significant application in the process industry. Its ability to handle constraints and multi-variable processes and its intuitive way of posing the control problem in the time domain are the reasons for its popularity. MPC problems with constraints are typically solved via quadratic programming using active set methods but these methods are known to scale badly with problem size. Developments in optimization, and especially in primal-dual interior point methods, have produced a new set of algorithms that are more efficient for large problems.

This thesis report the first known application of Krylov subspace methods to the sparse linear system in MPC. The results reported in this thesis show that the interior point method with Krylov subspace technique scales linearly with the prediction horizon N. Also, the implementation of the Krylov subspace iterative technique to active set method reduces the cost of computation from $O(N^3)$ to $O(N^{\frac{1}{2}})$.

This thesis is dedicated to the Almighty God

My ever present Help, my Hope for years to come

Acknowledgements

I would like to express my profound gratitude to Dr. Edward S Meadows, my supervisor, for his great supervision, encouragement and constant support during the entire course of my master's program. His ingenuity, experience, prudence and promptness have been an inestimable asset for me.

Dr. Fraser Forbes expressed his interest in my work and supplied me with his optimization texts and imparted the knowledge of process optimization on me through his optimization course, which have been useful tools for my research.

Dr. Huang and Dr. Meadows shared with me their knowledge of system identification and advanced process control. The knowledge you have imparted has been an invaluable one.

I had the pleasure of working with the Computer Process Control group of the University of Alberta. They are wonderful people and their support makes research like this possible. Special appreciations goes to my dear friends and colleagues: Samuel, Lekan, Folarin, Saheed, Folake, Omolara, Janna, Yinka and others for their friendship and encouragement.

I wish to thank my present and past office partners: Misha, Vinay, Amar, Monjur, Rouyu, Ian, Wesley, Sadiq and Tunde for creating a stimulating environment.

I give my thanks to the entire faculty and staff of Department of Chemical and Material Engineering for their help and the resources they made available for me.

Of course, I am grateful to my family for their patience and *love*. My heartfelt thanks go to my dearest sister, Yetunde for being so wonderful, you are more than a mother to me.

Finally, I wish to express my whole hearted appreciation and sincere gratitude to my loving husband, Olukayode for his unconditional love, unwavering support and affection. Without you this work would never have come into existence (literally). Thanks for making my living worthwhile, I will always love and cherish you.

Contents

1 INTRODUCTION 1

1.1 Scope and Organization of Thesis 3

1.2 Review of Model Predictive Control 4

1.3 Linear MPC Optimization Problem Formulation 6

1.3.1 Receding Horizon Formulation 6

1.3.2 Infinite Horizon Problem 7

1.3.3 Finite Horizon Problem 8

1.3.4 Problem Constraints 9

1.4 Quadratic Programming 9

1.5 Methods of Solving MPC Problem 11

1.5.1 Active Set Method 11

1.5.2 Interior Point Method 13

1.6 Review on Existing Work on QP approaches for MPC 15

1.7 Review on Sparse QP Solvers 17

2 ACTIVE SET METHODS 19

2.1 Introduction 19

2.2 Overview of active set QP solvers 20

2.2.1 QPOPT 21

2.2.2 QPKWIK 22

2.2.3 QPSchur 22

2.2.4 Active set Codes developed for this work 23

2.3 Hot Starting and Preprocessing 25

2.4 Null Space Active Set Method 26

2.4.1 Obtaining a basis for the null space of a matrix 27
2.4.2 Summarized Major Operations for Null-space 28
2.5 Range Space Active Set Method 29
2.5.1 Summarized Major Operations for Range-space 30
2.6 Direct Inversion of the Sparse KKT Equation 31
2.7 Krylov Subspace Iterative Method 32
2.8 Active Set Method Case Studies Application 37
2.8.1 Two Tank System 37
2.8.2 Shell Heavy Oil Fractionator 40
2.9 Conclusion 56

3 INTERIOR POINT METHODS **58**
3.1 Introduction 58
3.2 Path Following Method 59
3.3 Mehrotra Predictor-Corrector Method 61
3.4 Overview of IP Approaches for QP 62
3.4.1 LOQO 62
3.4.2 Riccati Recursion with IP 64
3.4.3 IP method Code developed for this work 64
3.5 Improvements on the Prototypical IP Method 65
3.5.1 Initial Point 65
3.5.2 Augmented System 65
3.6 Application of IP Method to Case Studies 66
3.6.1 Two Tank System 66
3.6.2 Shell Heavy Oil Fractionator 68
3.7 Conclusions 68

4 MATLAB OPTIMIZATION TOOLBOX **74**
4.1 Introduction 74
4.1.1 Large-Scale Algorithm 75
4.1.2 Medium-Scale Algorithm 75
4.2 Case Studies 76

4.2.1 Two Tank System 76
4.2.2 Shell Heavy Oil Fractionator Column 76
4.3 Conclusions 81

5 COMPLEXITY STUDIES **82**
5.1 Introduction 82
5.2 Feasibility and Constraints Softening 82
5.3 Dependence on N and H_u 83
5.3.1 Off-line Computation Time 84
5.3.2 Flops Count 84
5.3.3 Theoretical Cost Analysis 91
5.4 Numerical Results 92
5.5 Conclusions 93

6 CONCLUSIONS **95**
6.1 Contributions of Thesis 96
6.2 Recommendations For Users 97
6.3 Recommendations For Future Work 99

Bibliography **100**

A Mathematical Summary **104**
A.1 DEFINITIONS 104
A.1.1 Convexity of a Feasible Region (Domain) 104
A.1.2 Convexity of a Function 105
A.1.3 Linear Independence 105
A.1.4 Degrees of Freedom 105
A.2 SINGULAR VALUE DECOMPOSITION AND QR FACTORIZATION 106
A.2.1 Singular Value Decomposition 106
A.2.2 Condition Number 106
A.2.3 QR Factorization 106
A.3 NEWTON'S METHOD FOR SYSTEMS OF NONLINEAR ALGEBRAIC EQUATION 107

B DEVELOPED SOFTWARE **108**

B.1 Activesetnullmpc 108

B.2 Activesetrangempc 108

B.3 Activesetbackmpc 109

B.4 Activesetconjumpc 109

B.5 Interiormpc 109

C INPUT SCRIPT FILE **110**

List of Tables

1 Nomenclature for MPC Problem Formulation 15
2 Nomenclature for Optimization 16

2.1 Common Krylov Subspace Methods 33
2.2 Transfer function From Control Inputs to Outputs 46
2.3 Transfer function From Disturbances to Outputs 46
2.4 Uncertainty in the Gains of the Model 47
2.5 Steady State values of the Manipulated Variables 49
2.6 Steady State Values of the Controlled Outputs 49
2.7 Prototype Test Cases 50
2.8 Numerical Results for the Reduced space Method 52
2.9 Numerical Results for the Full space Method 56

3.1 Numerical Results for the Interior Point Method 68

4.1 Numerical Results for the Optimization Toolbox Application 81

5.1 Flops count using Null Space Active-set Method 87
5.2 Flops count using Range Space Active-set Method 87
5.3 Flops count using Backslash Operator Method 87
5.4 Flops count using BI-CGS Method 88
5.5 Flops count using Primal-Dual IP Method 88

List of Figures

1.1 Illustration of the Receding Horizon Approach 7

2.1 Coefficient Matrix Sparsity Pattern 25
2.2 A Two Tank System 38
2.3 Step Response Plot of the Two Tank System 41
2.4 Simulation Results of the Two tank System Using Active Set Method 42
2.5 Heavy Oil Fractionator Column 44
2.6 Summarized Version of Shell Control Problem 44
2.7 Open-loop Step Response Plot of the Nominal System 48
2.8 Closed-Loop Response of Case 1 53
2.9 Simulation Results for Case II and III Using Active Set Method . . . 54
2.10 Simulation Results for Case IV and V Using Active Set Method . . . 55

3.1 Simulation Results of the Two tank System Using Interior Point Method 69
3.2 Closed-Loop Response of Case 1 70
3.3 Simulation Results for Case II and III Using Interior Point Method . 71
3.4 Simulation Results for Case IV and V Using Interior Point Method . 72

4.1 Simulation Results of the Two tank System Using Optimization Toolbox in *Matlab* 77
4.2 Closed-Loop Response of Case 1 78
4.3 Simulation Results for Case II and III Using Optimization Toolbox in *Matlab* 79
4.4 Simulation Results for Case IV and V Using Optimization Toolbox in *Matlab* 80

5.1 CPU Trends for the Two Tank System 85
5.2 CPU Trends for the Shell Heavy Oil Fractionator Column 86
5.3 Logarithmic Plots of the Flops counts vs Prediction Horizon 89

6.1 Decision Tree 98

A.1 Convexity of a Feasible Region 104
A.2 Convexity of a Function 105

Nomenclature

Table 1: Nomenclature for MPC Problem Formulation

Quantity	Symbol
Change in manipulated variable	Δu
Controlled variable vector	z
Control horizon	H_u
Input penalty weight	R
Manipulated variable vector	u
Model output vector	y
Model state vector	x
Model state-space matrices	A, B, C
Optimization variable	x, u
Output penalty weight	Q
Penalty weight on control increment	S
Prediction horizon	N
Proportional controller gain matrix for LQR	K
Sampling Time	T_s
Set-point trajectory for controlled output	z_s
Steady state input vector trajectory	u_s
Terminal penalty weight on state	$\overline{Q}$

Table 2: Nomenclature for Optimization

Quantity	Symbol
Active constraint matrix	M
Complementarity or Duality measure	μ
Centering parameter	σ
Coefficient matrix in linear constraint	A
Diagonal matrix of slack variables	T
Diagonal matrix of the dual variables	Λ
Distance moved along a search vector (step length)	α
Dual variable or inequality constraint lagrange multiplier	λ
Equality constraint matrix	F
Equality constraint lagrange multiplier	π
Gradient at iteration k	g_k
Hessian matrix	G
Inequality constraint matrix	C
Jacobian matrix	J, F
Jacobian matrix of constraints	A
Lagrangian	L
Number of independent equality constraints	m_e
Number of independent inequality constraints	m_i
Objective function	$q, \mathcal{I}$
Path shaping parameter	τ
Preconditioner	$\tilde{M}$
Primal variables	x, u
Relaxation factor	w
Reduced Hessian	H_R
Search direction	p
Slack or surplus variable	t_i
Total number of decision variable	n
Working set at iteration k	W_k

Chapter 1

INTRODUCTION

This work is a study of optimization approaches for solving large scale Model Predictive Control(MPC) problems. It involves control of linear processes with quadratic objectives subject to general linear constraints. The proposed algorithms used the following cost function:

$$\min_{x,u} \sum_{j=1}^{N-1} (x(k+j) - x_s)^T Q (x(k+j) - x_s) + \sum_{j=0}^{H_u-1} (u(k+j) - u_s)^T R (u(k+j) - u_s) + (x(k+N) - x_s)^T \overline{Q} (x(k+N) - x_s) \quad (1.1)$$

where Q and $\overline{Q}$ are symmetric positive semi-definite matrices and R is a symmetric positive definite matrix. The vector u represents process inputs, or manipulated variables and the vector x represents process states. The vectors u_s and x_s represent the setpoints for the process inputs and states respectively. The above Eq 1.1 is subject to the process constraints described in Section 1.3.4.

The exact definition of *large scale* has not been universally accepted. A system is sometimes said to be of large scale when the conventional techniques of system theory and control cannot be successfully applied to it (Chen 1984). Another definition has been that a system is large in scale which can be decoupled into a number of subsystems or can be collectively controlled by many controllers. Typical of large scale systems are multiple input-multiple output (MIMO) systems, systems with large time delays, interactions, uncertainties and multiple constraints. Large scale systems are identified with large number of equations describing the process model

and constraints, and the solution of these systems of equations generally require high computational efforts. Perhaps the most practical definition of large scale for MPC is that the problem cannot be solved within a single sampling period.

The fundamental problem of controlling large-scale system is to generate a signal u which is capable of driving the state x or output y of the system from one location to another in finite time without violating any of the process constraints. The desired input signal to the system is often generated through an advanced control technology known as Model Predictive Control(MPC).

MPC is widely used in process industries as an effective means to deal with complex multi-variable control problems with constraints. The main idea of MPC is to choose the control action by repeatedly solving online optimal control problem. At each sampling time, starting at the current state the optimization aims at minimizing a linear or quadratic open loop performance objective subject to linear constraints over a future horizon. The first move of this open loop optimal manipulated variable profile is then implemented. This procedure is repeated at each control interval with the process measurements used to update the optimization problem.

This class of control algorithms is also referred to as moving horizon control or receding horizon control. It has several advantages for application in chemical process control. Section 1.2 and 1.3 give the full description of MPC and its optimization problem formulation.

The linear MPC controller uses a discrete time linear state space model of the plant which can be obtained from basic physics, process identification techniques or time series analysis techniques and does not require a significant fundamental modelling effort. Also, multi-variable processes can be handled by superposition of the linear models[1]. MPC has found significant application to large scale systems with constraints, but a significant drawback of MPC is the computational load, which limits applicability to slow or small problems.

In this work, optimization of the open-loop performance objective is performed by quadratic programming with constraints on the manipulated and controlled variables. Section 1.4 describes a typical quadratic programming problem and the necessary

[1]Combination of several linear models into a single multi-input, multi-output linear model

conditions for an optimum solution. Complex large-scale MPC problems are solved using improved algorithms that exploit the problem sparsity with efficient numerical methods. The main two approaches we consider in detail involve the active set method and the interior point method. Both methods are able to exploit the sparsity of the MPC optimization problem, as they must to obtain a solution in a reasonable amount of time.

This work should increase the range of applicability of MPC to large-scale systems.

1.1 Scope and Organization of Thesis

The thesis is organized as follows. The basics of MPC are reviewed first to derive the quadratic programming problem which needs to be solved to determine the optimal control action. Current methods of solution were reviewed to suggest possible improvement on these methods for speed and efficiency of MPC application to large-scale systems.

In Chapter 2, modified active set method techniques are emphasized and implemented on MPC optimization problems. A Krylov subspace iterative technique is used to solve the linear equations efficiently at each iteration of the active set method. Also, matrix preprocessing is implemented to avoid linearly dependent rows and a good feasible initial point for iteration is obtained through a constrained linear least squares algorithm. Both the full and reduced space active set methods are investigated. In the later part of this chapter, the improved active-set algorithms are applied to MPC case studies and the simulation results are presented.

An interior-point method algorithm is presented in Chapter 3 with more emphasis on *Mehtrotra's predictor-corrector* algorithm described by Mehrotra (1992*b*) and Wright (1997*c*). The Krylov subspace iterative technique is implemented on the linear systems from the predictor and corrector steps. The improved interior point algorithm is then applied to solve two process control problems.

Chapter 4 focuses on the *Matlab* Optimization Toolbox applied to large-scale MPC problems. The simulation results obtained from this application are then compared with the results of the improved active-set and interior-point methods discussed earlier.

Computational complexity results are presented in Chapter 5 to evaluate the performance of each method of solution as the MPC problem becomes larger.

The contribution of this thesis to model predictive control and the field of optimal control are highlighted and recommendations for users of the proposed optimization algorithms are presented in Chapter 6.

1.2 Review of Model Predictive Control

Model predictive control is a class of advanced control that uses a plant model to predict the effect of an input profile on the evolving state of the plant (Muske and Rawlings 1993).

There have been many developments in the field of optimal control. Bellman, in 1957 applied dynamic programming to optimal control of discrete-time systems (Bellman 1957). In 1958, Pontryagin and co-workers (Pontryagin *et al.* 1962) developed *maximum principle* to solve optimal control problems.

In the early 1960's Kalman published three major papers on optimal control. In 1960, Kalman designed a Linear Quadratic Regulator(LQR) controller for discrete-time systems (Kalman 1960*a*). He obtained an optimal gain matrix K such that the state-feedback law

$$u(k) = -Kx(k)$$

minimizes the cost function:

$$J = \sum_{j=0}^{\infty} (\|x(k+j)\|_Q^2 + \|u(k+j)\|_R^2) \tag{1.2}$$

subject to the state dynamics

$$x(k+1) = Ax(k) + Bu(k) \tag{1.3}$$

where the quadratic terms in the cost function are defined as follows:

$$\|x\|_Q^2 = x^T Q x \tag{1.4}$$

The vector u represents process inputs, or manipulated variables; vector x represents process states. The state vector is defined such that knowing its value at time k and

future inputs allows one to predict how the plant will evolve for all future time. Much of the power of Kalman's work relies on the fact that this general process model was used.

The cost function in Eq 1.2 penalizes squared input and state deviations from the set-points which are taken to be zero for simplicity of exposition. It includes separate tuning parameters, Q and R for state and input weight matrix respectively.

The infinite prediction horizon of the LQR algorithm endowed the algorithm with powerful stabilizing properties; it was proved to be stabilizing for any reasonable linear plant (stablizable and detectable) as long as the cost function weight matrices Q and R are positive definite. A dual theory was later developed to estimate plant states from noisy input and output measurements, using what is now known as a Kalman Filter, this is described in (Kalman 1960*b*) and (Kalman and Bucy 1961). The combined LQR controller and Kalman filter is called a Linear Quadratic Gaussian (LQG) controller. Constraints on the process inputs, states and outputs were not considered in the development of LQG theory. Although LQG theory provides an excellent and powerful solution to the problem of controlling an unconstrained linear plant, it had little impact on control technology development in the process industries. The most significant of the reasons for this failure include the following:

- constraints
- process nonlinearities
- model uncertainty (robustness)
- lack of user defined performance criteria

This failure of LQG led to the development in industry of a heuristic approach to solving on-line model-based control optimization problem. Input profiles of a process are computed to optimize future plant behavior over a prediction horizon. In the general case, process input and output constraints are included directly in the problem formulation so that future constraints violations are anticipated and prevented. The first input of the optimal input sequence is sent into the plant and the open loop problem is solved again at the next time interval using updated process measurements. In addition to developing more flexible control technology, new process identification

technology was developed to allow quick estimation of empirical dynamic models from test data, substantially reducing the cost of model development. This new methodology for industrial process modeling and control is what we now refer to as Model Predictive Control(MPC) technology. In summary, MPC, more than any other advanced control scheme, is more applicable in the process industry because of the following reasons:

- excellent ability to handle multi-variable processes
- excellent for systems with long time delays, interactions and constraints
- easy to understand because of the time domain definition of control problem
- takes account of actuator limitations and safety constraints

1.3 Linear MPC Optimization Problem Formulation

The discrete system model used by the controller is the state-space formulation shown in Eq 1.7. The model provides equality constraints on the user defined objective function. Actuator limitations, safety constraints or product specifications are inequality constraints. The objective function usually involves minimization of the squared norm of error between the predicted controlled outputs and a (vector) reference trajectory. The problem is defined to either be an infinite horizon or finite horizon problem. As described by Rawlings and Muske (1993) and Meadows *et al.* (1995) the infinite horizon consideration is required to ensure stability of the system.

1.3.1 Receding Horizon Formulation

The presence of constraints in MPC makes it difficult to obtain the closed loop formulation of the optimization problem. Most MPC applications solve the open-loop optimization on-line using a receding horizon technique. The receding horizon technique as described by Muske and Rawlings (1993) and Morari and Lee (1999) involves the following steps:

1. At time k and for current $x(k)$ solve an open-loop (OL) problem over a finite horizon.

2. Apply the first step of the resulting optimal OL control sequence.

3. Move the horizon to the next sampling time $k+1$ and repeat the procedure.

a graphical depiction of the above procedures is shown in Figure 1.1.

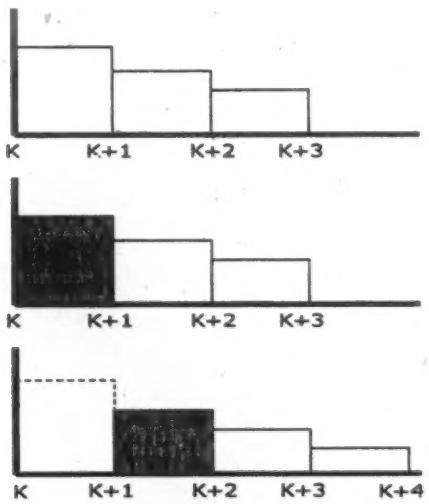


Figure 1.1: Illustration of the Receding Horizon Approach

1.3.2 Infinite Horizon Problem

The infinite open loop formulation is based on the minimization of the quadratic cost function given in Eq 1.5 below.

$$\min_{u} \sum_{j=0}^{\infty} \{y(k+j)^T Q y(k+j) + u(k+j)^T R u(k+j) + \Delta u(k+j)^T S \Delta u(k+j)\} \quad (1.5)$$

Q is a symmetric positive semi-definite (SPSD) penalty weight on the open-loop output y. R is a symmetric positive definite penalty weight on the open-loop input u. S is a SPSD penalty weight on the control increments $\Delta u(k+j) = u(k+j) - u(k+j-1)$. In this thesis the matrix S is set to zero.

1.3.3 Finite Horizon Problem

When the horizon is infinite, the number of decision variable is also infinite. Finite horizon problem is defined to give a finite parameterization for the open-loop minimization and to aid constraint handling. The finite horizon open-loop objective shown below:

$$\begin{aligned}\Phi_k &= \min_{x,u} x(k+N)^T \overline{Q}\, x(k+N) + \sum_{j=1}^{N-1}(x(k+j)^T C^T Q C\, x(k+j)) \\ &+ \sum_{j=0}^{H_u-1}(u(k+j)^T R\, u(k+j) + \Delta u(k+j)^T S\, \Delta u(k+j)^T) \qquad (1.6)\end{aligned}$$

in which H_u and N are the control and prediction horizon, respectively and $H_u \leq N$. The output penalty term in Eq (1.5) has been replaced with the corresponding terminal state penalty term in Eq (1.6). The terminal state penalty, $\overline{Q}$ depends on the plant model and the weighting matrix Q (Muske and Rawlings 1993). The function $x(k+N)^T \overline{Q}\, x(k+N)$ is called the terminal cost while the remaining part of the objective function is known as the trajectory cost. The terminal cost incorporates the infinite horizon solution into the problem. See Muske and Rawlings (1993) for a complete derivation.

The vector u contains the H_u future open-loop control moves as shown below.

$$u = \begin{bmatrix} u(k) \\ u(k+1) \\ \vdots \\ u(k+Hu-1) \end{bmatrix}$$

At time $k+H_u$, the manipulated variable $u(k+j)$ is set to zero for all $j \geq H_u$ in the open-loop objective function value calculation.

The receding horizon regulator uses the vector u that optimizes the open-loop objective function in Eq 1.6. The first input value in u, $u(k)$, is then injected into the plant. This procedure is repeated at each successive control interval with feedback incorporated by using the plant measurements to update the state vector at time k.

1.3.4 Problem Constraints

Both finite and infinite receding horizon problem are subject to constraints:

Model constraints (equality constraints):

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k) && k = 0, \ldots, H_{u-1} \\ x(k+1) &= Ax(k) + Bu(H_u - 1) && k = H_u, \ldots, N-1 \\ y(k) &= Cx(k) && k \geq 0 \end{aligned} \tag{1.7}$$

Input and output constraints (inequality constraints):

$$\begin{aligned} u^{min} \leq & u(k+j) \leq u^{max}, && j = 0, \ldots, H_u - 1 \\ y^{min} \leq & y(k+j) \leq y^{max}, && j = j_1, j_1 + 1, \ldots, j_2 \end{aligned} \tag{1.8}$$

in which:

$x \in \mathbb{R}^{n_x}$: vector of state variables

$u \in \mathbb{R}^{n_u}$: vector of manipulated variables

$y \in \mathbb{R}^{n_y}$: vector of process outputs

$u^{min} \in \mathbb{R}^{n_u}$: lower bound on manipulated variable

$u^{max} \in \mathbb{R}^{n_u}$: upper bound on manipulated variable

$y^{min} \in \mathbb{R}^{n_y}$: lower bound on process output

$y^{min} \in \mathbb{R}^{n_y}$: upper bound on process output

1.4 Quadratic Programming

The above MPC formulation fits into the general form of a convex quadratic programming (QP) problem as shown below:

$$\begin{aligned} \min_x q(x) &= \frac{1}{2} x^T G x + x^T d \\ s.t & \\ Fx &= g \\ Cx &\leq h \end{aligned} \tag{1.9}$$

in which:

$G \in \mathbb{R}^{n \times n} \geq 0$: SPSD matrix containing the penalty weights

$d \in \mathbb{R}^{n}$: vector containing the setpoint values

$x \in \mathbb{R}^{n}$: vector of the decision variable (inputs and states)

$F \in \mathbb{R}^{m_e \times n}$: coefficient matrix of the equality constraints (model equation)

$g \in \mathbb{R}^{m_e}$: equality constraints function values

$C \in \mathbb{R}^{m_i \times n}$: coefficient matrix of the inequality constraints (actuator limitations, safety constraints or product specifications)

$h \in \mathbb{R}^{m_i}$: inequality constraints function value

The above QP is convex as long as the Hessian matrix G is symmetric positive semi-definite (Gill *et al.* 1983). The necessary conditions for optimality of the convex QP in Eq 1.9 are that there exist π^* and λ^* such that $(x, \pi, \lambda) = (x^*, \pi^*, \lambda^*)$ and;

$$\begin{aligned} Gx + F^T\pi + C^T\lambda + d &= 0 \\ -Fx + g &= 0 \\ -Cw + h &\geq 0 \\ \lambda &\geq 0 \\ \lambda_j(-Cw + h)_j &= 0,\ j = 1, 2, \ldots, m_i \end{aligned} \tag{1.10}$$

where m_i is the number of rows of matrix C

The above equations are known as the *Karush-Kuhn-Tucker (KKT) optimality conditions.* The solution to Eq 1.10 becomes unique when the optimization problem is strictly convex, *i.e.,* when G is symmetric positive definite.

By introducing a vector t of slack variables for the inequality constraints, the KKT conditions become

$$F(x, \pi, \lambda, t) = \begin{bmatrix} Gx + F^T\pi + C^T\lambda + d \\ -Fx + g \\ -Cx - t + h \\ T\Lambda e \end{bmatrix} = 0 \tag{1.11}$$

$$(\lambda, t) \geq 0$$

where T and Λ are diagonal matrices defined by

$$\begin{aligned} T &= diag(t_1, t_2, \ldots, t_m) \\ \Lambda &= diag(\lambda_1, \lambda_2, \ldots, \lambda_m) \\ e &= (1, 1, \ldots, 1)^T \end{aligned}$$

It should be noted that the above KKT equation would be a set of linear equation except for the last term $T\Lambda e$.

1.5 Methods of Solving MPC Problem

The two approaches we consider in detail involve

1. the feasible-interior-point method
2. the active set method.

Both methods are able to exploit the large, sparse matrix structure in the MPC problem through efficient numerical algorithms. To exploit sparseness, this thesis provides a new technique based on Krylov subspace methods described by Trefethen and Bau (1997) and Golub and Loan (1996) to solve the key linear algebra problem in MPC calculations.

1.5.1 Active Set Method

This method solves a sequence of equality-constrained problems of the form,

$$\min_x \{q(x) : Ax = b\}$$

Given a feasible x_k, these methods find a step direction p_k by solving the sub-problem

$$\min_p \left\{q(x_k + p_k) : a_i^T(x_k + p_k) = b_i, i \in W_k\right\} \tag{1.12}$$

where q is the same as Eq 1.9, W_k is the working set[2] and a_i is the i-th row vector in the matrix A (Fletcher 1987). Expressing the sub-problem in terms of the step p_k

[2]The set of all the equality constraints and the active inequalities at iteration k

we define

$$\begin{aligned} p_k &= x - x_k \\ g_k &= , Gx_k + d \end{aligned}$$

and by substituting for x in the general QP problem we find that

$$\begin{aligned} q(x) &= q(x_k + p_k) = \frac{1}{2} p_k^T G p_k + g_k^T p_k + c \\ &s.t \\ a_i^T(x_k + p) &= b_i \end{aligned} \tag{1.13}$$

where $c = \frac{1}{2} x_k^T G x_k + d^T x_k$ is a constant term. The condition $a_i^T x_k = b_i$ is also satisfied at the k-th iteration. Since we can drop the constant c from the objective function, the QP sub-problem to be solved at the k-th iteration is as follows:

$$\begin{aligned} q(p) &= \min_p (\frac{1}{2} p_k^T G p_k + g_k^T p_k) \\ &s.t \\ a_i^T p_k &= 0 \text{ for all } i \in W_k \end{aligned} \tag{1.14}$$

When the optimal p_k is nonzero, we need to do a line search to decide how far to move in the obtained direction without violating any of the inactive constraints. We set

$$x_{k+1} = x_k + \alpha_k p_k$$

where the step length parameter α_k is chosen to be the largest value in the range $[0, 1]$ for which all the inactive constraints are satisfied. The maximum step length as described in (Gill *et al.* 1983) is given as the equation below.

$$\alpha_k \triangleq \min \left(1, \min_{i \notin I, a_i^T p_k < 0} \frac{b_i - a_i^T x_k}{a_i^T p_k} \right) \tag{1.15}$$

The iteration continues in this manner until we reach a point $\hat{x}$ such that p_k is within a tolerance of the origin. At this point, the quadratic objective function is minimized over its current working set $\hat{W}_k$. In order to obtain the optimal solution, the KKT described in Section 1.4 must be satisfied.

Karush-Kuhn-Tucker Conditions

The KKT optimality conditions for the QP sub-problem in Eq (1.14) are obtained as

$$\begin{bmatrix} G & a_i^T \\ a_i & 0 \end{bmatrix} \begin{bmatrix} p \\ \lambda \end{bmatrix} = \begin{bmatrix} -g_k \\ 0 \end{bmatrix} \tag{1.16}$$

since $p = 0$ satisfies the optimality condition then we have that

$$\sum_{i \in W_k} a_i \hat{\lambda}_i = G\hat{x} + d \tag{1.17}$$

for some Lagrange multipliers $\hat{\lambda}_i$, $i \in \hat{W}_k$. We need to examine the signs of the multipliers corresponding to the inequality constraints in the working set. If these multipliers are all nonnegative then we conclude that the optimal $\hat{x}$ is reached.

An active set method algorithm for solving convex quadratic programming problem is presented in Algorithm (1).

1.5.2 Interior Point Method

Applying interior point method involves solving the system of equations given in Eq 1.11 iteratively using Newton's method and carrying out a line search to enforce the positive orthant constraints, $(\lambda, t) \geq 0$. The development in this thesis follows that of Wright (1997*a*) and Wright (1997*c*). The IP algorithms stay in the strict interior of the feasible region by following a central path through the feasible region. The *central path* C is the set of points $(x_\tau, \lambda_\tau, t_\tau)$ parameterized by τ such that

$$F(x_\tau, \lambda_\tau, t_\tau) = \begin{bmatrix} 0 \\ 0 \\ \tau e \end{bmatrix}, \ (\lambda_\tau, t_\tau) > 0. \tag{1.18}$$

The *central path* strategy keeps the iterates in the interior by replacing $T\Lambda e = 0$ in Eq 1.11 with $\tau e = \sigma \mu e$ where $\sigma \in [0, 1]$ is the centering parameter and μ is the complementarity gap. The *complementarity gap* is a measure of the average value of the products $\lambda_i t_i$ and is defined as

$$\mu = \lambda^T t / m_i \tag{1.19}$$

Choose a feasible starting point x_0

Set the working set, W_0 to be a sub-set of the active constraints at x_0

Start iteration

Set the maximum iteration, $kmax$

Set $k = 1$

while $k < kmax$ **do**

  solve Eq 1.14 to find the step length, p_k

  if $p_k = 0$ **then**

    compute Lagrange multipliers $\hat{\lambda}_i$ that satisfy the Eq (1.17)

    set $\hat{W} = W_k$ *i.e.,* no blocking constraints

    if $\hat{\lambda}_i \geq 0$ *for all* $i \in W_k \cap I$ (*i.e.,* all inequalities constraints in the W_k) **then**

      stop with the optimal solution $x^* = x_k$

    else

      set $j = \arg\min_{j \in W_k \cap I} \hat{\lambda}_j$

      $x_{k+1} = x_k$; $W_{k+1} \leftarrow W_k \setminus \{j\}$ (*i.e.,* remove the constraints with the most negative value of $\hat{\lambda}_i$)

    end if

  else if $p_k \neq 0$ **then**

    compute the step length parameter, α_k from Eq (1.15)

    $x_{k+1} \leftarrow x_k + \alpha_k p_k$

    if there are blocking constraints **then**

      obtain W_{k+1} by adding one of the blocking constraints to W_k

    else

      $W_{k+1} \leftarrow W_k$

    end if

  end if

  $k = k + 1$

end while

Algorithm 1: Active-Set Method for Convex QP

where m_i is the number of bound constraints, *i.e.*, inequality constraints. Here $\sigma = 1$ corresponds to a pure centering direction, *i.e.*, moving all the pairwise products to the current average value μ, while $\sigma = 0$ corresponds to a standard Newton step. Various path following methods differ in the way the search direction is modified and σ is chosen. These steps are described in more details in Chapter 3.

Algorithm (2) shows the steps required for solving convex quadratic programming problem via an Interior Point method.

Choose *itermax* and *tol* and $\sigma_k \in [0, 1]$ and initialize with $k = 0$
Calculate $\mu_k = \lambda_k^T t_k / m_i$
if $\mu_k \leq tol$ or $k > itermax$ **then**
  STOP
end if
while $\mu_k > tol$ or $k < itermax$ **do**
  Solve the system of equation

$$\begin{bmatrix} G & F^T & C^T & 0 \\ -F & 0 & 0 & 0 \\ C & 0 & 0 & I \\ 0 & 0 & T & \Lambda \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta \pi \\ \Delta \lambda \\ \Delta t \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ -T_k \Lambda_k e + \sigma_k \mu_k e \end{bmatrix} \tag{1.20}$$

  Set $\alpha_k \in [0, 1]$ that retains $(\lambda_{k+1}, t_{k+1}) > 0$
  Set $(x_{k+1}, \pi_{k+1}, \lambda_{k+1}, t_{k+1}) = (x_k, \pi_k, \lambda_k, t_k) + \alpha_k(\Delta x, \Delta \pi, \Delta \lambda, \Delta t)$
  Set $k = k + 1$
end while

Algorithm 2: Primal-Dual Interior Point Method for Convex QP

1.6 Review on Existing Work on QP approaches for MPC

Quadratic programming (QP) is the vital heart of MPC applications in industry. There have been many studies published on tailoring efficient QP algorithms for online

implementations of MPC. A structured interior-point algorithm described by Mehrotra (1993) was applied to quadratic programs generated from MPC with linear model by Rao *et al.* (1998). This approach applies a discrete time Riccati transformation to the KKT system in the predictor and corrector steps at each iteration of the IP method. This approach guarantees growth of order of $O(N)$ of the computational cost with the prediction horizon compared with cubic growth for the active set approach. The author suggested that the use of sparse matrix solver for the KKT system would have increased computational performance. The resulting transformation leads to dense matrix decomposition that are $O(n_x^3)$, in which n_x is the number of state variables. The effectiveness of this approach was demonstrated by Rao *et al.* (1998) through simulations on a Polymer film process.

In 1999, Albuquerque *et al.* (1999) incorporated an interior point QP method with a successive quadratic programming (SQP) framework to develop **ISQP**. The **ISQP** uses the Mehrotra's Predictor Corrector algorithms in (Mehrotra 1993), but with a different choice of the centering parameter σ. **ISQP** performance was compared with two active set-based codes, **MINOS** and **QPSOL**, the results showed that **ISQP** converges in a fixed number of iterations and interior point methods provide great promise for the solution of MPC problem with a large number of inequality constraints. Biegler and Sentoni (2000) emphasized advanced interior point strategies for solving nonlinear MPC problem with inequality constraints and variable bounds. He shows that efficient IP methods preserve the particular problem structure and scale well in performance for large-scale problems in Nonlinear Model Predictive Control (NMPC). In a subsequent study by the same group, a Newton-type NMPC control algorithm was implemented on-line to control the liquid level and temperature in a pilot plant reactor in (Santos *et al.* 2001).

Later in 2002, Bartlett *et al.* (2002) developed **QPSchur**, which is an infeasible active set method implementing a Schur Complement algorithm. The MPC problem formulation in (Bartlett *et al.* 2002) involves only the control moves Δu as the decision variable and this results to a banded structure of the Hessian matrix for control horizon $H_u = 1$. **QPSchur** exploited this banded structure efficiently as compared with other QP solvers, **QPOPT**, **LOQO** and **QPKWIK**. However, the Hessian matrix losses its banded structure when $H_u \geq 2$. The algorithm was implemented

on-line on a paper machine process and it was concluded that the computational cost grows linearly $O(n_u)$ with the problem size for the banded system but increases to $O((n_u)(m))$ or even to $O((n_u^2)(m))$ when the number of active constraints becomes larger or for dense systems ($H_u \geq 2$). It should be noted that, despite its title, the reported results were applied only to the paper machine, and may not be applicable for other MPC problems.

1.7 Review on Sparse QP Solvers

The problem of solving large sparse linear systems has been a key issue in quadratic programming. Many researchers have tackled this problem to find a possible lasting solution.

Coleman and Li (1996) proposed a reflective Newton method for the minimization of a quadratic function of many variables subject to upper and lower bounds on some of the variables. The algorithm was the basis for the large scale algorithm in *Matlab* for convex QP with bounds constraints. Coleman and Li (1996) performed experiments on moderately large and sparse QP problems implementing the reflective Newton method. This algorithm involves applications of both sparse Cholesky factorization and preconditioned conjugate gradient method to the linear systems in the Newton step. Though, this method applies to general(indefinite) quadratic functions but it is only applicable to sparse QP problems with bounds constraints and not inequality.

A homogenous and self-dual algorithm for solving sparse monotone complementarity problem (MCP) described by Andersen and Ye (1995) was proposed by Andersen and Ye (1998). The self-dual algorithm was described as a single phase IP method, which was specialized for large-scale sparse convex optimization problems with nonlinear inequality constraints. The underlying linear algebra in the homogenous and self-dual algorithm involves application of an undefined matrix factorization method to a reduced Newton equations system (this is similar to the symmetric-quasi definite system in Vanderbei and Shanno (1999)). The homogenous and self-dual algorithm was implemented on quadratically constrained QP and geometric programming problems and the numerical results in Andersen and Ye (1998) showed that the algorithm is practically efficient for problems with large constraints (more than 100,000) and

variables.

Arioli (2000) performed a roundoff error analysis of a null space method for solving QP minimization problems. He combined the use of a direct QR factorization of the constraints with an iterative solver on the corresponding null space. The conjugate gradient method (CG) was used to solve the linear system in the null space algorithm when $(n-m)$ is large (the number of constraints is small compared to the unknowns) and Cholesky factorization was implemented for small $(n-m)$. Based on the numerical results obtained by Arioli (2000), the author concluded that the iterative algorithm performed efficiently on sparse matrices. Later in Arioli (2001), a similar analysis was performed with the combination of direct LU factorization of constraints with the CG iterative solver on the corresponding null space method. The results from Arioli (2001) gave evidence of the good performance of iterative solvers on sparse matrices.

A recent paper on the solution of sparse convex QP with bound constraints was presented by D'Apuzzo and Marino (2003). The paper focussed on parallel implementations of an IP algorithm to sparse convex QP with bound constraints. The solution of the linear system at each step of the IP method was carried out through an iterative approach based on the conjugate gradient method with block diagonal preconditioning technique. Incomplete Cholesky factorization of the Newton's equations was performed to obtain a preconditioner for the iterative step. The author performed experiments on randomly generated very sparse problems and showed that the inner iterative approach implementations obtained a constant CPU time reduction as the number of processor used increases.

In this work, a Krylov subspace iterative method (Bi-conjugate gradient stabilized method) with an incomplete LU factorization preconditioning step is implemented on QP problems from MPC. The efficiency of this implementations on the prototypical active set and interior point methods of solving QP problem from MPC is demonstrated through process control simulations.

Chapter 2

ACTIVE SET METHODS

2.1 Introduction

Active set methods are efficient in MPC applications if the active constraints rarely change at every iteration. These methods convert inequality constrained QP problem to linear equality constrained problem which requires solving systems of linear algebraic equations.

The various active-set method algorithms differ in the search direction calculation. Some methods involve matrix factorization and efficient update of factors at each iteration. Other methods, exploit the sparse matrix structure in the QP subproblem. In order to understand the key ideas presented in this chapter it may be helpful to first read Section A.1 and A.2.

As described in Section 1.5 active set method converts multi-variable quadratic programming (mv-QP) MPC problems to linear equality-constrained convex QP problems.

The active constraints must first be preprocessed to ensure linear independence. Starting with a feasible iterate x_k (computed for this work using *Matlab* function *initialpoint*) and the working set W_k, we first check whether x_k minimizes the objective function and satisfies the KKT conditions. If not, the method computes the step direction, p_k by solving an equality-constrained QP subproblem in which the constraints corresponding to the working set, W_k are regarded as equalities and all other constraints are temporary discarded. This iteration continues until the optimal

solution, x^* is obtained. An optimal solution is guaranteed since the QP problem is convex (Gill *et al.* 1983).

The necessary conditions for an optimum of a convex QP problem which is also known as the KKT optimality conditions, are (Gill *et al.* 1983):

1. Constraint Qualification: The rows of the active set matrix, M are linearly independent, *i.e* M has a full row rank

2. Feasibility : The active set is exactly satisfied at the optimum, x^*

$$Mx^* - b_M = 0$$

3. Stationarity: The gradient of the Lagrangian is stationary

$$\nabla_x L(x^*, \lambda^*) = \underline{0}$$

4. Non-Negativity:

$$\underline{\lambda}_N^* = 0, \ \underline{\lambda}_M^* \geq 0 \tag{2.1}$$

In which M is the Jacobian matrix associated with m linearly independent active constraints, $\underline{\lambda}_N$ and $\underline{\lambda}_M$ represent the column vectors of the Lagrange multipliers associated with inactive and active constraints respectively. For large scale MPC problem, the coefficient matrix is usually very large and sparse with nonzero entries in predictable positions. It is not diagonally dominant because of the zeros in the main diagonal. Section 2.2 gives the overview of the available active set method-based QP solver and highlights the new contributions made as part of this work.

2.2 Overview of active set QP solvers

For review, the QP solver **QPOPT**, **QPKWIK**, and **QPSchur** were selected because as described by Bartlett *et al.* (2002), each solver represents a very different approach of solving QP via active set method. The codes developed in this work are target to resolve the shortfalls of the reviewed solvers for efficient MPC applications.

2.2.1 QPOPT

QPOPT is a primal space QP solver implementing null space active set algorithm developed by Gill *et al.* (1995). It first finds a feasible initial point and updates this point through a reduced space active set method. Here, let $\hat{A} \in \mathbb{R}^{n \times n}$ be a matrix containing the Jacobian of the active constraints. **QPOPT** computes and updates a QR factorization of $\hat{A}^T$

$$\hat{A}^T = \begin{bmatrix} Y & Z \end{bmatrix} \begin{bmatrix} R \\ 0 \end{bmatrix} \tag{2.2}$$

in which:

n: number of the optimization decision variable

m: number of active constraints

$Y \in \mathbb{R}^{n \times m}$: orthogonal matrix

$Z \in \mathbb{R}^{n \times n-m}$: orthogonal basis matrix for the null space of $\hat{A}$

For general problem, the cost of performing the QR factorization in Eq 2.2 is $O(n(m^2))$ Schmid and Biegler (1994). Once the QR factorization is performed, **QPOPT** computes the reduced Hessian matrix $H_R = Z^T H Z$ through matrix-vector products of H with the columns of Z (which cost $O(n^2)$ for the dense Hessian). Cholesky factors of the reduced Hessian ($O(n-m)^3$) are then calculated and a back substitution is performed to compute the solution to the QP subproblem and give the estimate of the current active set. As **QPOPT** updates its active set the QR factors and the Cholesky factors are updated in an efficient manner. It was shown by Bartlett *et al.* (2002) that the cost of solving a QP with **QPOPT** scales as

$$O(n(m^2)) + O(n(n-m)) + O((n-m)^3).$$

Therefore if there are few active constraints, the cost of formulation $O(n(n-m))$ and Cholesky factorization $O((n-m)^3)$ will be expensive. The only case where **QPOPT** is expected to be efficient is when there is a good guess of the active set (so that the number of iteration is small) and the number of active constraints large (Bartlett *et al.* 2002).

2.2.2 QPKWIK

QPKWIK presented in (Schmid and Biegler 1994) is obtained from restructuring of the **ZQPCVX** code of Powell (1983) and is based on the method of Goldfarb and Idnani (1983). It involves QR factorization of a matrix related to $(U^H)^{-T}\hat{A} = QR$ where $(U^H)^{-1}$ is the inverse of the Cholesky factor of the Hessian matrix H and $\hat{A}$ is the gradient of the active constraints. The method does not exploit any structure of the Hessian matrix and also involves dense storage of the constraints Jacobian but sets up an internal sparse access to the Jacobian of the constraints so as to reduce the computation time. The cost of obtaining the initial inverse Cholesky factor of H is $O(n^3)$ and the cost per QP iteration is approximately $O((n^2)m)$ as reported in Bartlett *et al.* (2002). Therefore, the total cost of using **QPKWIK** to solve QP scales approximately $O(n^3) + O((n^2)m)$ when there are few constraints (Bartlett *et al.* 2002).

2.2.3 QPSchur

QPSchur as described in Bartlett *et al.* (2002) is a dual space algorithm like **QPKWIK**. An unconstrained minimum of the QP problem is found first and violated constraints are added until the solution is feasible. The algorithm implemented Schur complement approach to the linear system in Eq 2.3 below.

$$\begin{bmatrix} K_o & \hat{V} \\ \hat{V}^T & \hat{W} \end{bmatrix} \begin{bmatrix} \hat{v} \\ \hat{z} \end{bmatrix} = \begin{bmatrix} f_o \\ \hat{d} \end{bmatrix} \tag{2.3}$$

In which, K_o and f_o represent submatrices and vectors, respectively, of the original KKT system for a prespecified set of active constraints, and $\hat{v}$ represents the primal and dual variables for this initial system. The matrices $\hat{V}$ and $\hat{W}$ and the right hand side $\hat{d}$ represent additions to the active set in terms of additional inequality constraints. The vector $\hat{z}$ represents values of the primal and dual variables corresponding to these additions. A Schur complement $\hat{S}$ is defined for the coefficient matrix in Eq 2.3 as

$$\hat{S} = \hat{W} - \hat{V}^T K_o^{-1} \hat{V}$$

and the Eq 2.3 is solved as

$$\hat{z} = \hat{S}^{-1}(\hat{d} - \hat{V}K_o^{-1}f_o)$$
$$\hat{v} = K_o^{-1}(f_o - \hat{V}\hat{z})$$

The only computational cost was reported as the cost of updating the factorization of $\hat{S}$. For unconstrained QP, the total cost of implementing **QPSchur** was described by Bartlett *et al.* (2002) as $O(n)$ and $O(n^3)$ for banded and dense systems respectively. However, when there are many active inequality constraints, **QPSchur** requires many iterations and the cost of computation is $O(nm) + O(m^3)$ for banded Hessian and $O((n^2)m) + O(m^3)$ for dense Hessian (Bartlett *et al.* 2002).

2.2.4 Active set Codes developed for this work

The codes used in this work are programmed in *Matlab* for easy monitoring of the computational process. The *profile* tool in *Matlab* is used to optimize m-files by tracking their execution time. For each function in the m-file, the profiler records information about execution time, number of calls, parent functions, child functions, code line hit count, and code line execution time (Mathworks 2002).

1. **Activesetnullmpc**

   Activesetnullmpc is a primal null space active set *Matlab* m-file developed as part of this work to solve QP problem from MPC. The proposed null space active set method QP solver is similar to **QPOPT** in that it implements reduced space active set method, working in the space containing only the degree of freedom $(n-m)$. **Activesetnullmpc** also applies a sparse constrained linear least square step (CLLS) described in Section 2.3 to obtain a good guess of the active set and perform efficient update of the QR factors at every iteration. Also, for efficient on-line MPC applications, the solution of the previous open-loop minimization is shifted one time forward, updated and use as a good approximation to the solution of the next QP. In this approach, on-line computation time decreases with time so also the number of iterations for the QP solution. This approach is described in detailed in Section 2.4. The theoretical cost per iteration of solving

QP with **Activesetnullmpc** scales as $O(n(m^2)) + O(n(n-m)) + O((n-m)^3)$ like **QPOPT**.

2. **Activesetrangempc**

   Activesetrangempc is a primal range space active set *Matlab* m-file developed as part of this work to solve QP problem from MPC. The proposed range-space active set method QP solver implements the range-space method described in (Gill *et al.* 1983), with the addition of hot-starting to obtain a good guess of the initial active set. It is similar to **QPSchur** developed by Bartlett *et al.* (2002) because it also uses a Schur Complement approach. But unlike **QPSchur** that starts with an infeasible point, **Activesetrangempc** is a feasible point method. It guarantees a feasible solution even if the optimization is terminated prematurely for any reason. Like **QPOPT**, it also implements QR factorization on the Jacobian of the active constraints but uses the space containing the constraints. Therefore, the theoretical cost of solving QP problem with **Activesetrangempc** is

   $$O(n(m^2)) + O(nm) + O(m^3)$$

   in which $O(n(m^2))$ is used for QR factorization like **QPOPT**, and $O(nm) + O(m^3)$ for reduced matrix formulation and Cholesky factorization. However, there is an additional off-line computational cost of finding the inverse of the Hessian matrix. This algorithm is expected to perform very well when there are few active constraints.

3. **Activesetconjumpc**

   The proposed **Activesetconjumpc** is a full space active set method *Matlab* m-file developed as part of this work to solve QP problem from MPC. It incorporates Krylov subspace iterative techniques described by Golub and Loan (1996) and Trefethen and Bau (1997) into the QP solution. This thesis reports the first known application of Krylov subspace method for solving linear algebra system in MPC.

   Figure 2.1 shows the typical sparsity pattern of the coefficient matrix in the KKT equations. In order to exploit this structure efficiently, a Krylov-subspace

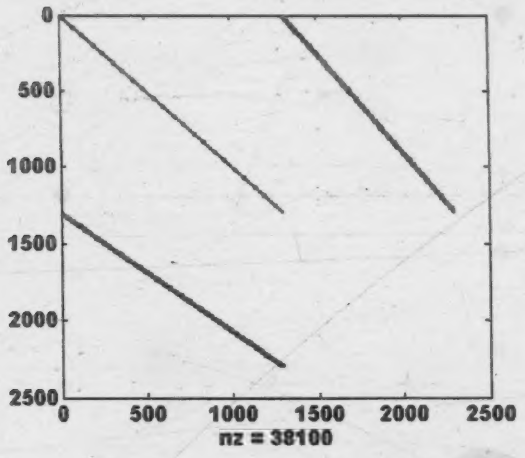


Figure 2.1: Coefficient Matrix Sparsity Pattern

iterative method (bi-conjugate gradient stabilized) described by Golub and Loan (1996) is implemented. Unlike **QPSchur**, this solver supports the use of large prediction and control horizons for MPC applications. The Krylov-subspace iterative methods is discussed in Section 2.7. The algorithm in **Activesetconjumpc** also incorporates a preconditioning step to prevent ill-conditioning of the coefficient matrix and to speeds up the convergence of the iterative method. This algorithm reduces the cost of solving the tested QP problems from $O(N^3)$ to $O(N^{\frac{1}{2}})$.

2.3 Hot Starting and Preprocessing

When an active set method is used to solve large-scale MPC problems, availability of a good feasible initial-point usually reduces the cost of computation and the problem is solved with fewer steps.

Hot start means that the initial point, the working set, Lagrange multipliers and the corresponding matrix factors that are obtained from the previous open-loop minimization are updated and used to initialize the next open-loop minimization. A good starting point is obtained using constrained linear least squares via function `lsqlin`

from *Matlab Optimization Toolbox* (Mathworks 1992). This minimization is shown below:

$$\begin{aligned} \min_{x} \quad & \tfrac{1}{2}\|Nx - b_N\|^2 \\ s.t \quad & \\ & Mx = b_M \\ & Nx \leq b_N \end{aligned} \tag{2.4}$$

in which:

$N \in \mathbb{R}^{m_i \times n}$: inequality Jacobian matrix

$x \in \mathbb{R}^n$: unknown vector (decision variable)

$M \in \mathbb{R}^{m_e \times n}$: equality Jacobian matrix

$b_M \in \mathbb{R}^{m_e}$: function value of the equality

$b_N \in \mathbb{R}^{m_i}$: function value of the inequality

m_e: number of equality constraints

m_i: number of inequality constraints

From Eq 2.4 above, the starting point for the active set method is chosen to be the value of x that minimizes the residual norm of the inequality constraints subject to both the equality and inequality constraints (Gill *et al.* 1983).

Preprocessing involves removal of redundant rows of the active constraints matrix to prevent linear dependence of the rows to satisfy the constraint qualification requirement for an optimum. The preprocessing involves checking if the rank of Jacobian matrix M. Rows of M associated with zero singular values are removed.

Hot starting and preprocessing is performed once in every QP minimization and the cost for this process decreases with time for the subsequent open-loop minimization because the result of the previously solved problem are incorporated in the next QP solution.

2.4 Null Space Active Set Method

Null space active set methods convert the linear equality-constrained QP subproblem in Eq 1.14 to an unconstrained problem by finding a full-rank matrix $Z \in \mathbb{R}^{n \times (n-m)}$

such that Z spans the null space of the coefficient matrix A (Gill *et al.* 1983). The feasible subspace for p_k is formed from a basis, Z_k whose columns are orthogonal to the estimate of the active set A_k (*i.e* $A_k Z_k = 0$). Substituting $p_k = Z_k t_k$ into Eq 1.14 gives

$$\min_t \frac{1}{2} t_k^T Z_k^T G Z_k t_k + g_k^T Z_k t_k \tag{2.5}$$

The optimality condition for the unconstrained problem becomes

$$\begin{aligned} Z_k^T G Z_k t_k + g_k^T Z_k &= 0 \\ t_k &= -(Z_k^T G Z_k)^{-1} Z_k^T g_k \end{aligned} \tag{2.6}$$

where $Z_k^T g_k$ is the projected gradient and $Z_k^T G Z_k$ is the projected Hessian matrix. Thus, the solution to the Eq (1.14) is obtained as:

$$\begin{aligned} p_k &= Z_k t_k \\ \lambda_k &= -A_k (A_k^T A_k)^{-1} (G p_k + g_k). \end{aligned} \tag{2.7}$$

2.4.1 Obtaining a basis for the null space of a matrix

An orthogonal basis for the null space of a matrix A can be obtained numerically using any of the following techniques:

1. Using *Matlab* function `null(A)`

2. Perform the singular value decomposition of the matrix

   $$[U, S, V] = \mathbf{svd}(A)$$

   and choose the columns of V corresponding to zeros in S

3. Obtain the orthogonal QR matrix factors of A^T

   $$\begin{aligned} [Q, R] &= \mathbf{qr}(A^T) \\ Q &= [YZ] \end{aligned}$$

   this follows that the first m columns of Q can be taken as the columns of Y (range space) and the last $(n - m)$ columns of Q as the columns of matrix Z (null space)

For MPC applications, the third technique is used for the computation of matrix Z. This is because the QR factorization allows update of the orthogonal matrix factors when columns are added or deleted from the active set during the iteration. This reduces the cost of computing the orthogonal factors from the scratch. See Gill *et al.* (1983) and Fletcher (1987) for the detailed explanation of the QR factors update procedures.

In summary, the step direction, p_k and the Lagrange multipliers, λ_k are computed using the null space active set method using the equations below:

$$\begin{aligned} t_k &= -(Z_k^T G Z_k)^{-1}(Z_k^T g_k) \\ p_k &= Z_k t_k \\ \lambda_k &= -R_k^{-1} Y_k^T (G p_k + g_k) \end{aligned} \tag{2.8}$$

It should be noted however that the step direction computation involves the inversion of the dense reduced Hessian matrix $Z_k^T G Z_k \in \mathbb{R}^{(n-m)\times(n-m)}$.

Section 2.4.2 shows steps for the null-space active set method. These are used to determine the total floating point operations required per iteration for the null-space active set method presented in 5.

2.4.2 Summarized Major Operations for Null-space

The major steps for the null space active set method are described below.

1. Calculation of the orthogonal matrix Z by QR factorization.This is performed only once for each open-loop minimization.

$$\begin{aligned} [Q, R] &= qr(A^T) \\ Z &= Q(:, m+1:n) \\ Y &= Q(:, 1:m) \end{aligned}$$

2. Matrix by matrix multiplication for calculating the projected Hessian matrix

$$Z_k^T G Z_k = Z_k^T \times G \times Z_k$$

3. Matrix by vector multiplication for the projected gradient

$$Z_k g_k = Z_k^T \times g_k$$

4. Backslash operation for t calculation

$$t_k = -Z_k^T G Z_k \setminus Z_k g_k$$

5. Matrix by vector multiplication for step direction, p_k calculation

$$p_k = Z_k \times t_k$$

6. Back-substitution for the Lagrange multipliers calculation

$$\lambda_k = -R_k^{-1} Y_k^T (G p_k + g_k)$$

7. Vector addition for updating the primal variables

$$x_{k+1} = x_k + \alpha_k p_k$$

2.5 Range Space Active Set Method

Range space methods use the block matrix inversion approach (Schur complement approach) described by (Gill *et al.* 1983) to solve the KKT equation in Eq 1.16 and reduce the dimension of the matrix needed to be inverted from $(n+m) \times (n+m)$ to $m \times m$ and also take advantage of orthogonal matrix factorization and updates for easy computation. The approach is described below:

$$\begin{aligned} M &= \begin{bmatrix} G & A_k^T \\ A_k & 0 \end{bmatrix} \\ M^{-1} &= \begin{bmatrix} Q & R \\ S & T \end{bmatrix} \end{aligned} \tag{2.9}$$

in which:

$$\begin{aligned} Q &= G^{-1} - G^{-1}A_k^T(A_kG^{-1}A_k^T)^{-1}A_kG^{-1} \\ R &= G^{-1}A_k^T(A_kG^{-1}A_k^T)^{-1} \\ S &= (A_kG^{-1}A_k^T)^{-1} \\ T &= -(A_kG^{-1}A_k^T)^{-1} \end{aligned} \tag{2.10}$$

Therefore, result to the following sets of equation for p_k and λ_k

$$\begin{aligned} \lambda_k &= -(A_kG^{-1}A_k^T)^{-1}A_kG^{-1}g_k \\ p_k &= -(A_k^T\lambda_k + g_k)^TG^{-1} \end{aligned} \tag{2.11}$$

The expression for the range space version of the step direction, p_k given in Eq 2.11 is not appropriate for practical computation (Gill *et al.* 1983). Let R_k be the $m_k \times m_k$ upper triangular matrix associated with the QR factorization of A_k^T and Y_k be the matrix whose columns are the first m_k columns of the orthogonal factor Q_k. Substituting the expression

$$A_k = [Y_k\ R_k]^T = L_kY_k^T$$

into the range-space equation, we obtain

$$\begin{aligned} p_k &= -G^{-1}(Y_k\theta + g_k) \\ \theta_k &= (Y_k^TG^{-1}Y_k)^{-1}Y_k^TG^{-1}g_k \\ \lambda_k &= -R_k^{-1}Y_k^T(Gp_k + g_k) \end{aligned} \tag{2.12}$$

The step direction calculation in Eq 2.12 involves inverting the Hessian matrix G once during the process and inverting $m \times m$ matrix $Y_k^TG^{-1}Y_k$ at each iteration.

Section 2.5.1 shows the steps for each iteration for the range-space active set method.

2.5.1 Summarized Major Operations for Range-space

The following steps describe the steps for the range space method. These steps are used for calculating the flops per iteration for range space active set method in Chapter 5.

1. Hessian matrix inverse calculation. This is performed once throughout the minimization operation

2. QR factorization of A_k^T performed once for each major iteration

$$\begin{aligned} [Q, R] &= qr(A^T) \\ Y_k &= Q(:, 1:m) \\ R_k &= R(1:m, :) \end{aligned}$$

3. Matrix by matrix multiplication for $Y_k^T G^{-1} Y_k$

$$Y_k^T G^{-1} Y_k = Y_k^T \times G^{-1} \times Y_k$$

4. Backslash operation for θ calculation

$$\theta_k = Y_k^T G^{-1} Y_k \setminus (Y_k^T G^{-1}) g_k$$

5. Matrix by vector multiplication for step direction,p_k calculation

$$p_k = -G^{-1} \times (Y_k \theta + g_k)$$

6. Back-substitution for the Lagrangian multiplier calculation

$$\lambda_k = -R_k^{-1} Y^T (Gp + g_k)$$

7. Vector addition for updating the primal variables

$$x_{k+1} = x_k + \alpha_k p_k$$

2.6 Direct Inversion of the Sparse KKT Equation

Null and range space methods are reduced space active set methods which focus on factorization of the Jacobian matrix of the active constraints in the KKT equations. The backslash operator in *Matlab* can solve the full space problem more efficiently by exploiting the sparse matrix property of the KKT equations.

If the coefficient matrix T is a square sparse matrix, then the backslash operation on T stores only the non-zeros of T with the indices and eliminates computations with zeros (Mathworks 1992).

Computation involves a choice between the following two algorithms:

1. If T is symmetric and has positive real diagonal elements, the backslash operator finds a symmetric minimum degree order p and attempts to compute the Cholesky factorization of $T(p,p)$. If successful, it finishes with two sparse triangular back-substitutions.

2. If T is not diagonally dominant, or if Cholesky factorization fails, the backslash operator finds a column minimum degree order p, computes the LU factorization with partial pivoting of $T(:,p)$ and performs two sparse triangular back-substitutions.

For the active set MPC problem, the matrix $T \in \mathbb{R}^{(n+m)\times(n+m)}$ is symmetric but not diagonally dominant. Therefore, the backslash operator in *Matlab* uses the second algorithm to compute the step direction, p_k and the Lagrangian multipliers, λ_k directly.

2.7 Krylov Subspace Iterative Method

Iterative methods are often more efficient than the direct methods when solving large sparse linear systems. Iterative methods generate a sequence of approximate solutions of the unknown variables x_k and only involve matrix by vector multiplication.

The stationary iterative methods, such as Gauss-Siedel, Jacobi and SOR methods are not applicable to MPC problem because they require the coefficient matrix to be diagonally dominant. A diagonally dominant matrix is a matrix A with the property that we have

$$|a_{ii}| \geq \sum_{j \neq i} |a_{ij}| \tag{2.13}$$

for all i (Golub and Loan 1996). The coefficient matrix of the KKT equations has zeros in the main diagonal as shown in Figure 2.1; therefore, non-stationary iterative methods are required for MPC applications.

Krylov subspace methods are an important class of iterative solvers for large sparse linear systems (Golub and Loan 1996). These methods are characterized by the subspaces in which the iterates lie. The i-th Krylov subspace for a given matrix, A and the residual vector r_0 associated with the initial point x_0 is

$$K_i(A, r_0) = \{r_0, Ar_0, A^2 r_0 \dots A^{i-1} r_0\}$$

The i-th iterate of a Krylov subspace method lies in the shifted subspace

$$x_i \in x_0 + K_i(A, r_0)$$

Krylov subspace methods differ in how a vector in the subspace is selected and how the new iterate is found. The most common Krylov subspace methods are shown in Table 2.1 (Golub and Loan 1996). The CG method select x_i by minimizing the

Table 2.1: Common Krylov Subspace Methods

Name	Acronym	Applications
Conjugate Gradient Method	CG	SPD squared matrix
Minimal Residual	MINRES	SID squared matrix
Generalized Minimal Residual	GMRES	General squared matrix
Bi-Conjugate Gradient	Bi-CG	General squared matrix
Conjugate Gradient Squared	CGS	General squared matrix
Bi-Conjugate Gradient Stabilized	Bi-CGSTAB	General squared matrix

matrix norm of the error over the Krylov subspace. It is only applicable to symmetric positive definite matrix.

MINRES selects x_i by minimizing the two-norm of the residual vector over the Krylov subspace, it is applicable to symmetric but possibly indefinite matrix.

GMRES is applicable to general square matrix, it selects x_i by minimizing the two-norm of the residual vector over the Krylov subspace. GMRES is known to increase complexity and storage space requirement linearly with each iteration.

BICG generates two CG-like sequences of vectors for the coefficient matrix, A and its transpose. It is useful when the coefficient matrix is nonsymmetric and nonsingular

however, minimization of residues is not guaranteed and it is known to have extremely erratic convergence behavior.

CGS is a variant of BICG that applies the updating operation of the coefficient matrix and its transpose to the same vector. This method sometimes gives unreliable results, although its convergence rate is twice as fast as BICG.

The Bi-CGSTAB method was developed to solve general linear systems while avoiding the often irregular convergence of the CGS. In most cases, it converges at the same rate as CGS and it can be viewed as a hybrid of CGS and GMRES.

The convergence rate of a Krylov subspace method can be improved by using the method in conjunction with a preconditioner which can be denoted by $\tilde{M}$. The iterative method is applied to the preconditioned system

$$\tilde{M}^{-1}Ax = \tilde{M}^{-1}b$$

This work reports use of a preconditioned Bi-CGSTAB method to solve the large, sparse systems of equations generated from the MPC problems. The main operations are

1. matrix vector product
2. dot product and norm computation
3. vector update

Algorithm (3) shows the algorithm for the pre-conditioned Bi-CGSTAB method.

Preconditioner Choice

The rate of convergence of an iterative method to solve $Ax = b$ typically depends on the distribution of the eigenvalues of A. Bi-CGSTAB converges faster if the spectrum is clustered, *i.e.*, there is little spread in the eigenvalues of A.

Preconditioning can be viewed as a transformation to a system $\tilde{A}\tilde{x} = \tilde{b}$ for which $\tilde{A}$ has a more favorable spectrum than A. Some of the simplest preconditioner are the following

Choose $itermax$ and tol and initialize with $i = 0$
Guess an initial solution $x^{(0)}$ of $Ax = b$
Compute $r^{(0)} = b - Ax^{(0)}$
Choose $\tilde{r} = r^{(0)}$ (shadow residual)
Set $i = 1$
while $i < itermax$ **do**
  $\rho_{i-1} = \tilde{r}^T r^{(i-1)}$
  if $norm(r) \leq tol$ **then**
    STOP
  end if
  if $\rho_{i-1} = 0$ **then**
    method fails
  end if
  if $i = 1$ **then**
    $p^{(i)} = r^{(i-1)}$
  else
    $\beta_{i-1} = (\rho_{i-1}/\rho_{i-2})(\alpha_{i-1}/\omega_{i-1})$
    $p^{(i)} = r^{(i-1)} + \beta_{i-1}(p^{(i-1)} - \omega_{i-1}v^{(i-1)})$
  end if
  Solve
  $M\hat{p} = p^{(i)}$
  $v^{(i)} = A\hat{p}$
  $\alpha_i = \rho_{i-1}/\tilde{r}^T v^{(i)}$ (step length)
  Set
  $x_{half} = x + \alpha_i \hat{p}$
  $r_{half} = b - A\, x_{half}$
  Check norm of r_{half}
  if small enough **then**
    Set
    $x = x_{half}$
    $iter = i - 0.5$
    STOP
  else
    $s = r^{(i-1)} - \alpha_i v^{(i)}$
    Solve
    $M\hat{s} = s$
    $t = A\hat{s}$
    $\omega_i = t^T s / t^T t$
    $x^{(i)} = x_{half}^{(i)} + \omega_i \hat{s}$
    $r^{(i)} = s - \omega_i t$
    Check convergence and continue if necessary. It is necessary that $\omega_i \neq 0$ to avoid stagnation
  end if
end while

Algorithm 3: The Preconditioned Bi-conjugate Gradient Stabilized Method

1. Symmetric Successive Over-relaxation(SSOR) preconditioner

$$\tilde{M} = \frac{1}{\omega(2-\omega)}(D+\omega L)D^{-1}(D+\omega L)$$

where ω is the relaxation factor(takes value between 0 and 2), D is the diagonal matrix of the diagonal element of matrix A , L is the lower triangular part of matrix A.

2. Sparse Incomplete LU Preconditioner

$$A \approx \tilde{L}\tilde{U}$$

$$\tilde{M} = \texttt{luinc}(A)$$

This step is performed in *Matlab* using the function `luinc`. As each column j of the triangular incomplete factors is being computed, the entries smaller in magnitude than a local drop tolerance are dropped from the appropriate factor. The only exceptions to this dropping rule are the diagonal entries of the upper triangular factor, which are preserved to avoid a singular factor (Mathworks 1992).

3. Sparse Incomplete Cholesky Preconditioner for symmetric positive definite matrix A

$$A \approx \tilde{R}^T\tilde{R}$$

$$\tilde{M} = \texttt{cholinc}(A)$$

The sparse incomplete Cholesky factorization for a symmetric positive definite matrix is performed in *Matlab* using the function `cholinc`.

The first alternative requires an optimal choice of the relaxation factor ω which can be obtained from the univariate optimization (Eq 2.14) step described in (Golub and Loan 1996).

$$\min_{\omega} \rho(M_\omega^{-1}N_\omega) \tag{2.14}$$

where

$$N_\omega = (1-\omega)D - \omega U$$

$$M_\omega = D + \omega L$$

Based on our testing, this step is too expensive for MPC applications. The third approach requires a symmetric positive definite matrix, but the equations to be solved have a semi-definite coefficient matrix. Therefore, the algorithm used in this work implements the second choice of preconditioner.

2.8 Active Set Method Case Studies Application

The effectiveness of the active set methods are demonstrated by applying it to a Two-Tank system and Shell Heavy Oil Fractionator Column MPC case studies problems presented in Section 2.8.1 and 2.8.2.

2.8.1 Two Tank System

In most large scale systems there are several interconnected components which usually give rise to high order dynamic responses. We wish to obtain a model of a two tank system and implement MPC to keep the states of the system to half of their initial values at steady state while satisfying a hard constraints on the input. A two tank system is shown in Fig 2.2.

Assumption:

- liquid in the tank is incompressible (density is constant)
- unsteady state, variable volume
- inert (no chemical reaction)
- connecting pipes have uniform cross-section

Mass Balance

$$\rho A_1 \frac{dh_1}{dt} = m_i - m_t \tag{2.15}$$

$$\rho A_2 \frac{dh_2}{dt} = m_t - m_e \tag{2.16}$$

$$m_t = \rho a\sqrt{2gh_1} \tag{2.17}$$

$$m_e = \rho a\sqrt{2gh_2} \tag{2.18}$$

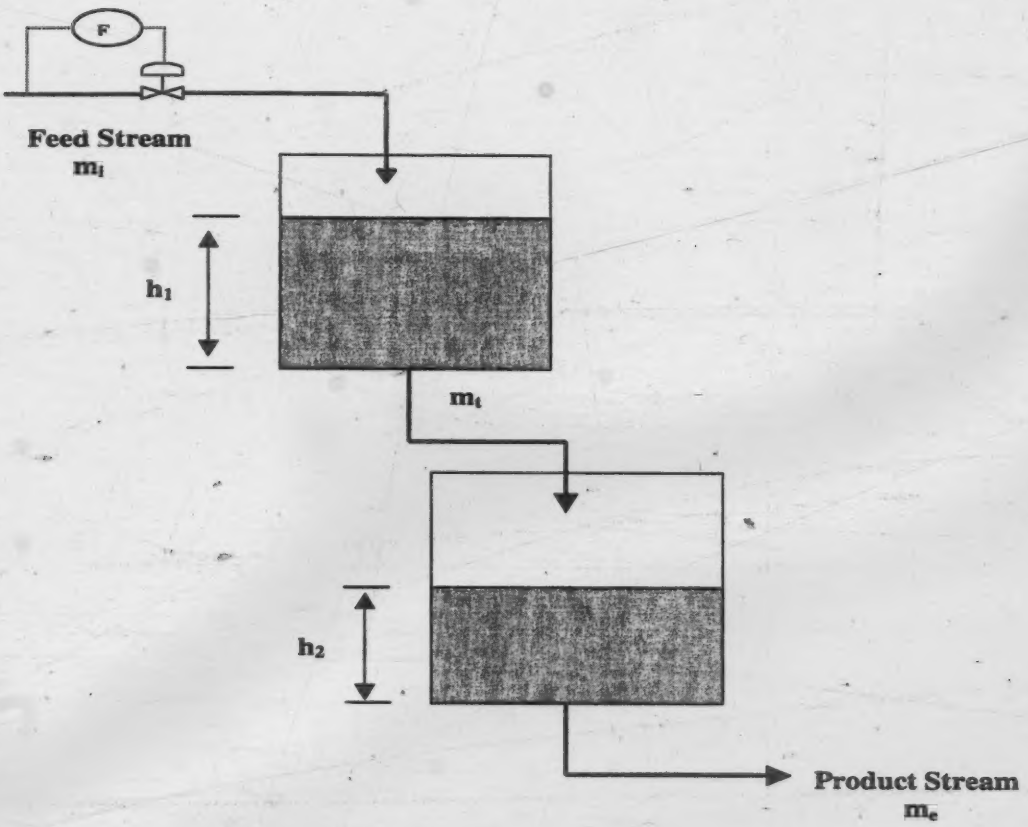


Figure 2.2: A Two Tank System

Substituting for m_t and m_e in Eqns (2.15) and (2.16), the two mass balance equations become

$$\frac{dh_1}{dt} = \frac{m_i}{\rho A_1} - \frac{a}{A_1}\sqrt{2gh_1} \tag{2.19}$$

$$\frac{dh_2}{dt} = \frac{a}{A_2}\sqrt{2gh_1} - \frac{a}{A_2}\sqrt{2gh_2} \tag{2.20}$$

The continuous time state space linearized model matrices of the two tank system is given below.

$$A = \begin{bmatrix} -\frac{a}{2A_1}\sqrt{\frac{2g}{h_{1s}}} & 0 \\ \frac{a}{2A_2}\sqrt{\frac{2g}{h_{1s}}} & -\frac{a}{2A_2}\sqrt{\frac{2g}{h_{2s}}} \end{bmatrix}, \; B = \begin{bmatrix} \frac{1}{\rho A_1} \\ 0 \end{bmatrix}, \; C = \begin{bmatrix} 0 & \frac{\rho a}{2}\sqrt{\frac{2g}{h_{2s}}} \end{bmatrix}, \; D = 0 \tag{2.21}$$

Therefore the transfer function of the input, m_i to output, m_e is obtained as

$$\begin{aligned} \frac{m_e(s)}{m_i(s)} &= C(sI - A)^{-1}B + D \\ G(s) = \frac{m_e(s)}{m_i(s)} &= \frac{K_1}{s^2 + K_2 s + K_1} \end{aligned} \tag{2.22}$$

where

$$K_1 = \frac{a^2 g}{2A_1 A_2}\frac{1}{\sqrt{h_{1s}h_{2s}}}, \qquad K_2 = \frac{a\sqrt{2g}}{2}\left(\frac{1}{A_2\sqrt{h_{1s}}} + \frac{1}{A_1\sqrt{h_{2s}}}\right)$$

Consider a two-tank system with a second order dynamics given below.

$$\frac{y(s)}{u(s)} = \frac{2}{s^2 + 3s + 2}, \;\; x(0) = [1, 1]$$

Figure 2.3 is the step response plot for the system. The plot shows that the system is asymptotically stable. Using a sampling time, $T_s = 0.1s$, the resulting discrete time state-space representation is shown below

$$x(k+1) = \begin{bmatrix} 1.7236 & -0.7408 \\ 1.0000 & 0 \end{bmatrix} x(k) + \begin{bmatrix} 0.1250 \\ 0 \end{bmatrix} u(k)$$

$$y(k) = \begin{bmatrix} 0.0724 & 0.0656 \end{bmatrix} x(k)$$

The control objectives are the following:

1. Setpoint at 0.5
2. Input bound, $-2 \leq u(k) \leq 2$

Cost Function

$$\min_{u,x}(x(k+N)-x_s)^T\,\overline{Q}\,(x(k+N)-x_s)+\sum_{j=1}^{N-1}(x(k+j)-x_s)^T Q(x(k+j)-x_s)$$
$$+\sum_{j=0}^{H_u-1}(u(k+j)-u_s)^T R(u(k+j)-u_s) \quad (2.23)$$

Constraints

1. Model Equations (Equality Constraint)

$$x(k+1+j)=\begin{bmatrix}1.7236 & -0.7408\\ 1.0000 & 0\end{bmatrix}x(k+j)+\begin{bmatrix}0.1250\\ 0\end{bmatrix}u(k+j),\ j=0\ldots N-1$$

$$u(k+j)\triangleq u(k+H_u-1),\ j=H_u\ldots N$$

2. Inequality Constraint

$$-2\leq u(k+j)\leq 2,\ j=0\ldots H_u-1$$

Using prediction horizon $N=5$ and control horizon $H_u=3$, the two states are assumed to be equally important thus choosing the penalty weights, $Q=diag([2,2])$ for the states and the penalty weight $R=0.1$ is chosen for the input. The steady state value of the input, u_s is obtained as 0.0690. The terminal penalty weight on the states, $\overline{Q}$ is obtained from discrete Lyapunov equation using the state matrix A and the penalty matrix Q. From the closed loop MPC response of the system shown in Figure 2.4 it can be concluded that the MPC controller design satisfies the control objectives without any constraints violation throughout the sampling period.

2.8.2 Shell Heavy Oil Fractionator

A heavy oil fractionator cracks crude oil into several oil fractions for further processing. Figure 2.5 shows a Shell oil fractionator column which was presented at the first Shell Process Control Workshop in 1987 (Prett and Morari 1987). Its linear model does not represent a specific plant, but was made to encapsulate the relevant control issues faced in a real oil fractionators. Three products streams are drawn off from the top, side and bottoms of the fractionator. Product specifications for the top and

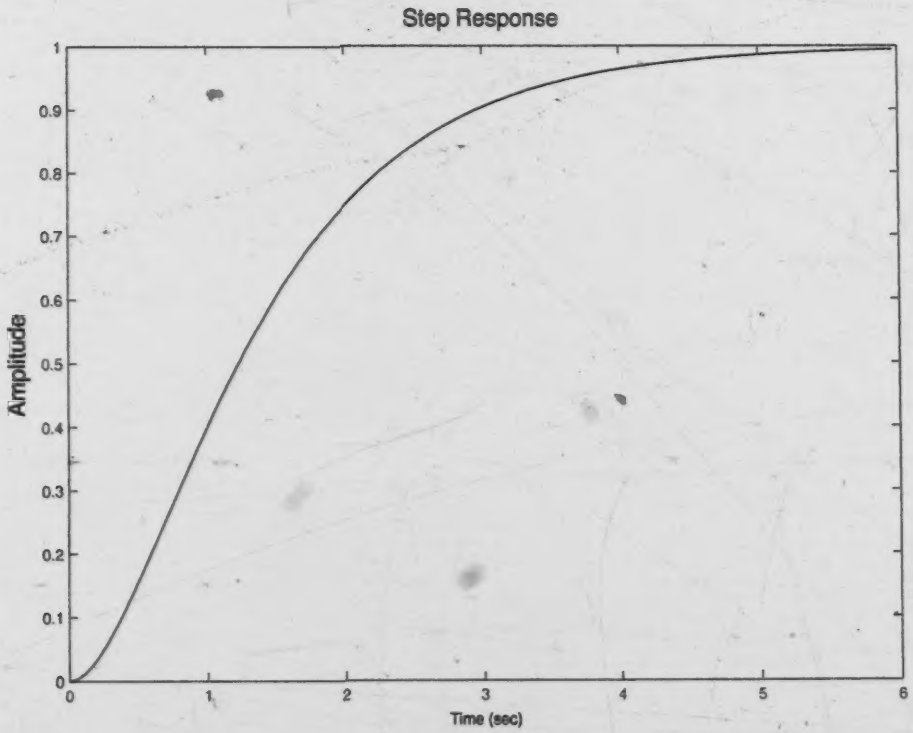
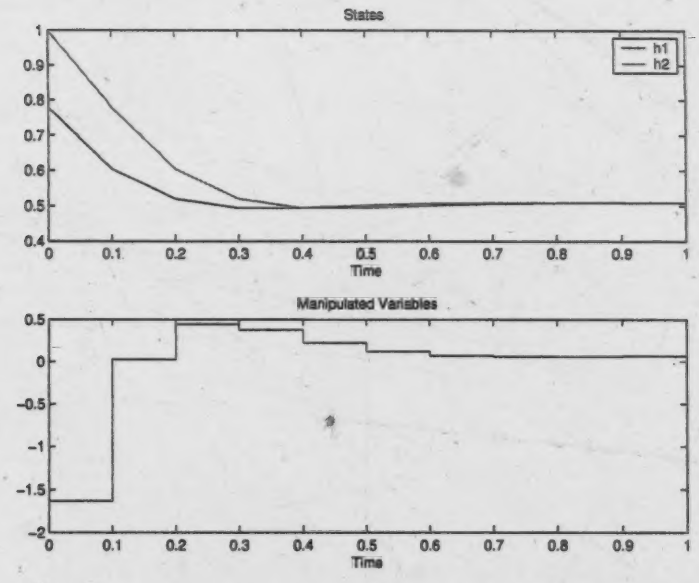
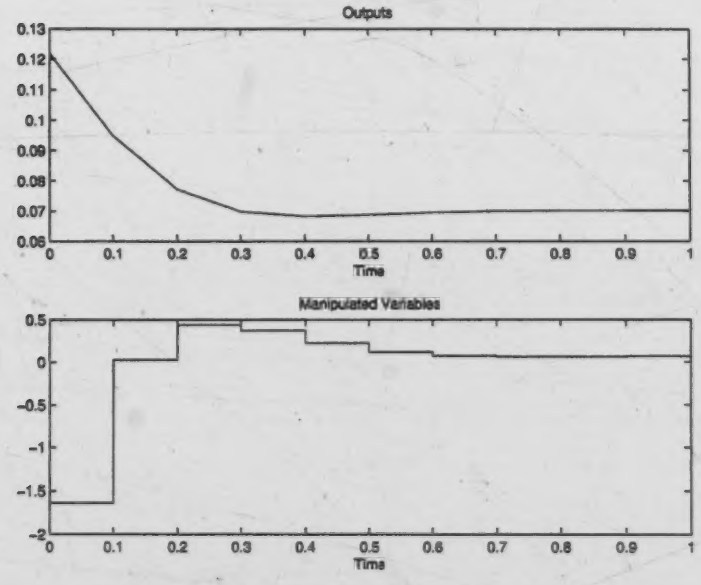


Figure 2.3: Step Response Plot of the Two Tank System



(a) Closed-Loop MPC Response for States and Input



(b) Closed-Loop MPC Response for Output and Input

Figure 2.4: Simulation Results of the Two tank System Using Active Set Method

side draws composition are determined by economics and operating requirements. There is no product specification for the bottom draw, but there is an operating constraints on the temperature in the lower part of the column. The three circulating reflux loops are used to remove heat from the column by means of heat exchangers to achieve desired product separation and supply heat to other processes in other part of the plant. The heat removed from the top two reflux loops is determined by the requirement of other processes and these therefore acts as unmeasured disturbance inputs to the fractionator. The bottom loop has an enthalpy controller which regulates heat removal in the loop by adjusting the steam make. Its heat duty is a manipulated variable, which can be used to control the process. The bottom reflux is used to generate steam for use on other units, therefore, though the bottom reflux duty can be used to control the column, there is need to keep it as low as possible to maximize the amount of steam generated and thus minimize cost.

The top and side draws are additional inputs. Altogether, there are five inputs to the fractionator of which three are manipulated inputs and two are unmeasured disturbances. There are seven measured outputs from the fractionator, the first two are the top and side end point compositions and the other five are temperature measurement for the column top, upper reflux, side draw, intermediate reflux and the bottom reflux. Only three of these are controlled outputs; the two compositions and the bottom reflux temperature. The remaining four outputs measurements are available for use by the controller. Figure 2.6 shows the summarized version of the heavy oil fractionator.

Control of a Heavy Oil Fractionator using Model Predictive Control

This case study intended to show how improved optimization techniques can be used to design an efficient and fast model predictive controller for a fairly complex control problem. The different active set algorithms are implemented to evaluate the numerical capability of each algorithm to solve the open loop convex QP problem associated with the control of Shell Heavy Oil Fractionator Column. This case study involves constraints handling, multi-variable control, optimization, and the sensitivity analysis of the controlled system to gain uncertainty and disturbances.

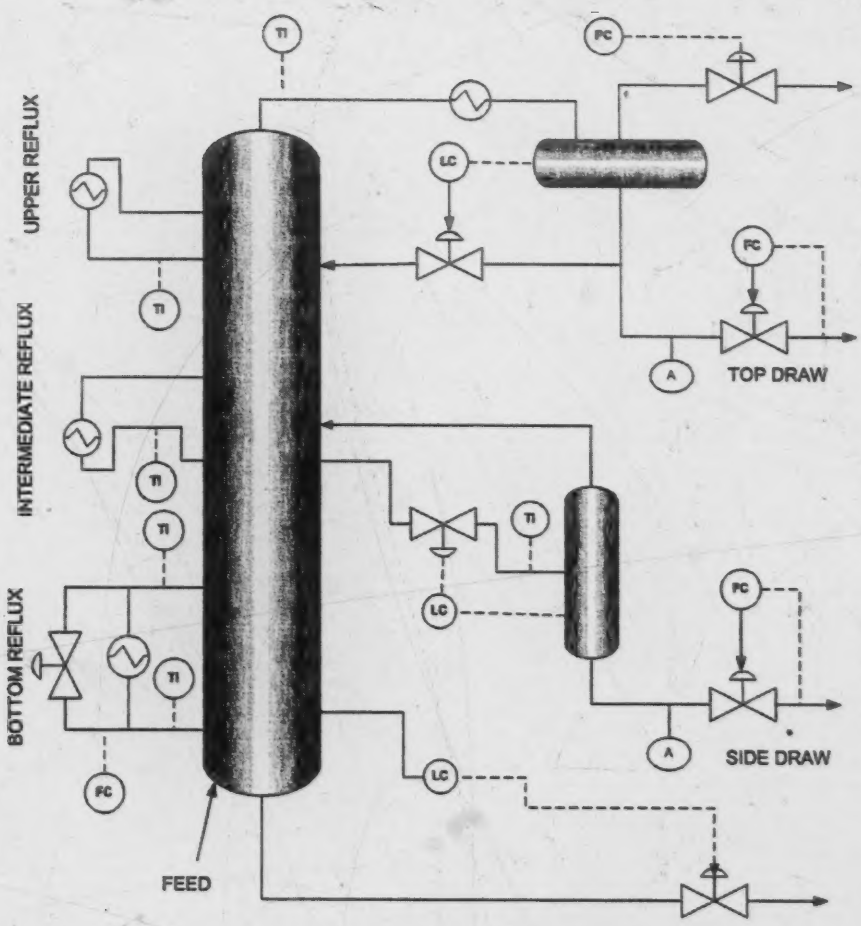


Figure 2.5: Heavy Oil Fractionator Column

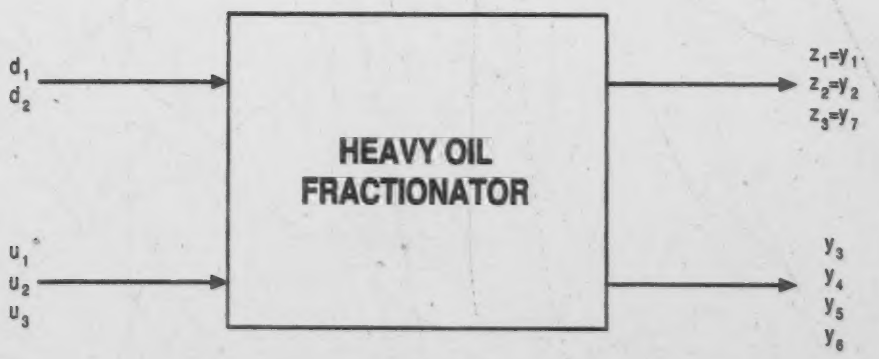


Figure 2.6: Summarized Version of Shell Control Problem

Control Objectives

1. Maintain the top and side draw product end points at specification (0.0 ± 0.005 at steady state).
2. Minimize the bottom reflux duty in order to maximize the heat recovery (*i.e* maximize steam generation).
3. Reject unmeasured disturbances entering the column from the upper and intermediate refluxes due to change in heat duty requirements from other columns (Upper and intermediate reflux duties range between -0.5 and 0.5). For disturbance rejection, the requirement is to return the top and side end point compositions to their set points, subject to satisfying constraints.
4. Keep the closed-loop speed of response between 0.8 and 1.25 of the open-loop process bandwidth.

Control Constraints

The following conditions must be met:

1. All draws must be within the hard constraints of 0.5 and -0.5.
2. The bottom reflux heat duty is constrained within the hard bounds of 0.5 and -0.5.
3. All manipulated variables have maximum move size limitations of magnitude 0.05 per minute.
4. Fastest sampling time is 1 minute.
5. The bottom reflux draw temperature has a minimum value of -0.5.
6. The top endpoint must be maintained within the maximum and minimum values of 0.5 and -0.5.

The input constraints are hard constraints and cannot be violated. The output constraints are to be met if possible or soften in case if infeasibility problem arises.

Process Model

Table 2.2 shows the nominal transfer functions from control inputs to all outputs (first order lag with time delays). Table 2.3 shows the nominal transfer functions from the disturbances to all outputs. Table 2.4 specifies the uncertainty in the gains of the model. The nominal plant is obtained when $\epsilon_j = 0$ for each j; but it is assumed that ϵ_j can vary in the range $-1 \leq \epsilon_i \leq 1$.

Table 2.2: Transfer function From Control Inputs to Outputs

	u_1	u_2	u_3
y_1	$4.05e^{-27s}\frac{1}{50s+1}$	$1.77e^{-28s}\frac{1}{60s+1}$	$5.88e^{-27s}\frac{1}{50s+1}$
y_2	$5.39e^{-18s}\frac{1}{50s+1}$	$5.72e^{-14s}\frac{1}{60s+1}$	$6.90e^{-15s}\frac{1}{40s+1}$
y_3	$3.66e^{-2s}\frac{1}{9s+1}$	$1.65e^{-20s}\frac{1}{30s+1}$	$5.53e^{-2s}\frac{1}{40s+1}$
y_4	$5.92e^{-11s}\frac{1}{12s+1}$	$2.54e^{-12s}\frac{1}{27s+1}$	$8.10e^{-2s}\frac{1}{20s+1}$
y_5	$4.13e^{-5s}\frac{1}{8s+1}$	$2.38e^{-7s}\frac{1}{19s+1}$	$6.23e^{-2s}\frac{1}{10s+1}$
y_6	$4.06e^{-8s}\frac{1}{13s+1}$	$4.18e^{-4s}\frac{1}{33s+1}$	$6.53e^{-1s}\frac{1}{9s+1}$
y_7	$4.38e^{-20s}\frac{1}{33s+1}$	$4.42e^{-22s}\frac{1}{44s+1}$	$7.20\frac{1}{19s+1}$

Table 2.3: Transfer function From Disturbances to Outputs

	d_1	d_2
y_1	$1.20e^{-27s}\frac{1}{45s+1}$	$1.44e^{-27s}\frac{1}{40s+1}$
y_2	$1.52e^{-15s}\frac{1}{25s+1}$	$1.83e^{-15s}\frac{1}{20s+1}$
y_3	$1.16\frac{1}{11s+1}$	$1.27\frac{1}{6s+1}$
y_4	$1.73\frac{1}{5s+1}$	$1.79\frac{1}{19s+1}$
y_5	$1.31\frac{1}{2s+1}$	$1.26\frac{1}{22s+1}$
y_6	$1.19\frac{1}{19s+1}$	$1.17\frac{1}{24s+1}$
y_7	$1.14\frac{1}{27s+1}$	$1.26\frac{1}{32s+1}$

The step response of the nominal system is shown in Figure 2.7. This plot shows

Table 2.4: Uncertainty in the Gains of the Model

	u_1	u_2	u_3	d_1	d_2
y_1	$4.05 + 2.11\epsilon_1$	$1.77 + 0.39\epsilon_2$	$5.88 + 0.59\epsilon_3$	$1.20 + 0.12\epsilon_4$	$1.44 + 0.16\epsilon_5$
y_2	$5.39 + 3.29\epsilon_1$	$5.72 + 0.57\epsilon_2$	$6.90 + 0.89\epsilon_3$	$1.52 + 0.13\epsilon_4$	$1.83 + 0.13\epsilon_5$
y_3	$3.66 + 2.29\epsilon_1$	$1.65 + 0.35\epsilon_2$	$5.53 + 0.67\epsilon_3$	$1.16 + 0.08\epsilon_4$	$1.27 + 0.08\epsilon_5$
y_4	$5.92 + 2.34\epsilon_1$	$2.54 + 0.24\epsilon_2$	$8.10 + 0.32\epsilon_3$	$1.73 + 0.02\epsilon_4$	$1.79 + 0.04\epsilon_5$
y_5	$4.13 + 1.71\epsilon_1$	$2.38 + 0.93\epsilon_2$	$6.23 + 0.30\epsilon_3$	$1.31 + 0.03\epsilon_4$	$1.26 + 0.02\epsilon_5$
y_6	$4.06 + 2.39\epsilon_1$	$4.18 + 0.35\epsilon_2$	$6.53 + 0.72\epsilon_3$	$1.19 + 0.08\epsilon_4$	$1.17 + 0.01\epsilon_5$
y_7	$4.38 + 3.11\epsilon_1$	$4.42 + 0.73\epsilon_2$	$7.20 + 1.33\epsilon_3$	$1.14 + 0.18\epsilon_4$	$1.26 + 0.18\epsilon_5$

$$-1 \leq \epsilon_i \leq 1;\ i = 1, 2, 3, 4, 5$$

that the system is asymptotically stable with settling times that range from about 10 minutes to about 200 minutes

Controller Design

Since the relevant open loop time constants range from 40 minutes to 60 minutes (obtained from the transfer functions of the manipulated variable to the end point compositions) ; closed-loop time constants are to be between 0.8 and 1.25 of these values. The sampling time is chosen as the 0.1 of the minimum open-loop time constant *i.e.*, $T_s = 4$ minutes ; the same value as chosen in some process control literatures (Prett and García 1988) and (Maciejowski 2002). Using the Control System Toolbox function `c2d` and `tf2ss` the continuous model is converted to a discrete time state space model with the redundant modes removed using the Toolbox's minimal realization function `minreal`. This model is presented in *Matlab* m-file *shelloil.m* and it is available for download from: http://www.ualberta.ca/~fadebiyi.

One of the objectives of the Shell Control Problem is to maximize the steam make in the steam generator *i.e.*, minimize the bottom reflux duty. In order to achieve this objective, the bottom reflux duty u_3 is made as an additional controlled output z_4 (with a one-step delay to ensure strictly proper system).

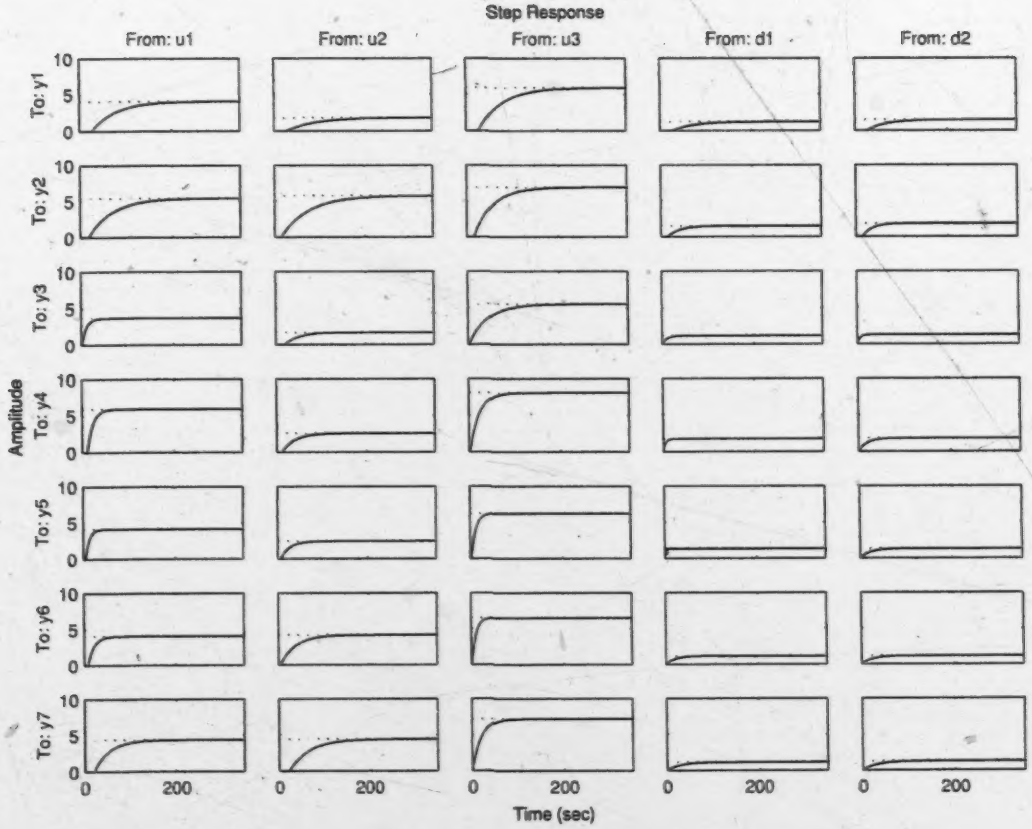


Figure 2.7: Open-loop Step Response Plot of the Nominal System

The steady state values for z_1 and z_2 are the specified set points (0.0 ± 0.005). There is need to keep z_4 as low as possible without constraints violation, therefore, to obtain the set point of z_3 and z_4, the steady state value of z_4 is minimized subject to satisfaction of all the constraints using linear programming. The steady state values of the manipulated variable and controlled outputs are computed and shown in Table 2.5 and Table 2.6 respectively. These steady state values are used as reference trajectory for the MPC applications.

Table 2.5: Steady State values of the Manipulated Variables

Manipulated Variable	Steady State Values
Top Draw u_1	0.5000
Side Draw u_2	-0.0872
Bottom Reflux Duty u_3	-0.3280

Table 2.6: Steady State Values of the Controlled Outputs

Controlled Outputs	Steady State Values
Top End Point, z_1	0.0000
Side Endpoint z_2	0.0000
Bottom Reflux Temperature z_3	-0.5000
Bottom Reflux Duty z_4	-0.3280

Prototype Test Cases

Table 2.7 shows the five cases representing different disturbances and gain uncertainties. The following conditions must be met.

- $-0.5 \leq u_1, u_2, u_3 \leq 0.5$
- $-0.5 \leq z_1, z_2, z_3 \leq 0.5$
- $|\Delta u_1|, |\Delta u_2|, |\Delta u_3| \leq 0.05 T_s$

- $T_s \geq 1$ minute

Table 2.7: Prototype Test Cases

Case	d_1	d_2	ϵ_1	ϵ_2	ϵ_3	ϵ_4	ϵ_5
I	0.5	0.5	0	0	0	0	0
II	−0.5	−0.5	−1	−1	−1	1	1
III	−0.5	−0.5	1	−1	1	1	1
IV	0.5	−0.5	1	1	1	1	1
V	−0.5	−0.5	−1	1	0	0	0

The objective is to regulate z_1 and z_2, the top and side draw end points, to their set points while minimizing u_3 in order to maximize the heat recovery and satisfy the above constraints (Prett and Morari 1987). The inputs constraints are hard constraints and cannot be violated. In order to obtain a feasible starting point a constraint window was set for the output constraints.

Reference Tracking

Since we are interested in computing the future states x and manipulated variable u such that the predicted controlled outputs z coincides with its reference trajectory z_s, therefore, the cost function is defined as a convex quadratic function that penalizes deviations of the predicted variables from their set-point values.

Cost Function

We define the cost function to be

$$\begin{aligned} \mathcal{I} &= \min_{x,u} \Big(\sum_{i=1}^{N-1} (Cx(k+i) - z_s)^T Q (Cx(k+i) - z_s) + \sum_{i=0}^{H_u-1} (u(k+i) - u_s)^T R (u(k+i) - u_s) \\ &+ (Cx(k+N) - z_s)^T \overline{Q} (Cx(k+N) - z_s)\Big) \end{aligned} \tag{2.24}$$

where z_s is the desired set-point value of the controlled output vector and Q is the SPSD weighting matrix for the deviation of the predicted output from z_s, u_s is the

desired value of the controlled input vector at steady state and R is a symmetric positive definite weighting matrix for the deviation of the predicted input vector from u_s. The terminal penalty weight on the controlled output, $\overline{Q}$ is obtained from the discrete Lyapunov equations and ensure stability of the system (Maciejowski 2002).

Constraints

The constraints on the Shell problem are defined as

Model Equations (Equality Constraints)

$$\begin{aligned} x(k+1+i) &= Ax(k+i) + Bu(k+i) && i = 0, 1, \ldots, H_{u-1} \\ x(k+1+i) &= Ax(k+i) + Bu(k+H_u-1) && i = H_u, \ldots, N-1 \end{aligned} \tag{2.25}$$

Inequality Constraints

Hard Constraints

$$\begin{aligned} -0.5 \leq\ & u(k+i) \leq 0.5 && i = 0, 1, \ldots, H_u - 1 \\ -0.05T_s \leq & \Delta u(k+i) \leq 0.05T_s && i = 0, 1, \ldots, H_u \end{aligned} \tag{2.26}$$

Soft Constraints

$$-0.5 \leq Cx(k+i) \leq 0.5 \qquad i = H_w \ldots N,\ H_w \geq 1 \tag{2.27}$$

where N is the prediction horizon, H_u is the control horizon and H_w is the controlled outputs constraints window.

Simulation Results

Using prediction horizon $N = 20$ and control horizon $H_u = 20$, we assumed constant weights over the prediction horizon. All the manipulated variables are assumed to be equally important except for the top draw which tends to hits upper constraints quite often. The first two controlled output are the most important outputs. Based on the assumptions, we choose $Q = diag[2,\ 4,\ 0.2,\ 0.001]$ and $R = diag[0.6,\ 0.3,\ 0.3]$.

Table 2.8 and 2.9 show the open-loop numerical results for the reduced and full space active set methods. The closed-loop simulations results for all the prototype test cases presented in Table 2.7 are shown in Figure 2.8, 2.9 and 2.10.

Table 2.8– 2.9 depict the numerical results for all the prototype test cases of the Shell control problem presented in Table 2.7. Table 2.8 shows that the reduced space methods (**Activesetnullmpc** and **Activesetrangempc**) requires more CPU for all the tested cases when compared with the results of the Krylov subspace iterative active set method (**Activesetconjumpc**) presented in Table 2.9. It is also shown that the number of iterations ranges between 2 and 3 as a result of the initial hotstarting incorporated at the onset of the minimization process.

Figure 2.8 through 2.10 illustrate the high sensitivity of the bottom reflux temperature z_3 to the two disturbances. For example, in Case 1 there are positive step changes in the two disturbances. This should increase d_1 and d_2 and reduce the input u_3 from the bottom reflux loop to the column. Figure 2.8 shows an initial decrease in the bottom reflux temperature z_3 as a result of decrease in u_3 until when it almost hits its lower bound -0.5 and then, u_3 starts to increase so as to increase z_3 and prevent constraint violation.

Table 2.8: Numerical Results for the Reduced space Method

Problem	n	nbnb	Null Space			Range Space		
			nfn	CPU(s)	iter	nfn	CPU(s)	iter
Case 1	260	0	39	0.9930	2	39	1.1601	2
Case 2	260	0	40	1.0335	2	39	1.2266	3
Case 3	260	0	36	1.0051	2	36	1.2218	3
Case 4	260	0	36	0.9750	2	36	1.1677	2
Case 5	260	0	39	0.9758	2	39	1.1705	2

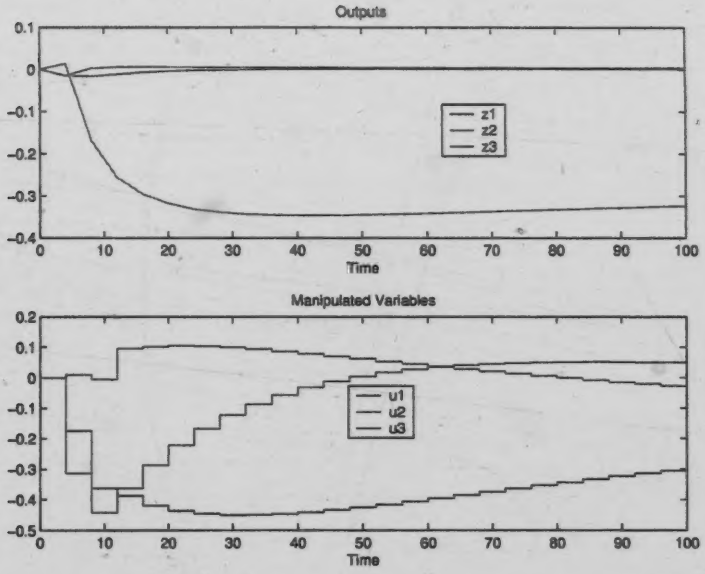
where,

n = number of variables

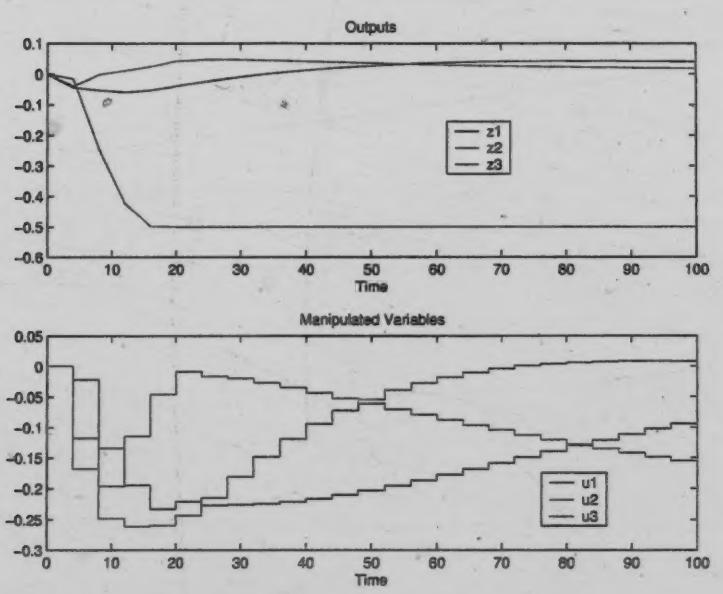
nbnb = number of active bounds at the solution

nfn = the total number of functions evaluations

iter = total number of iteration for the first open loop minimization

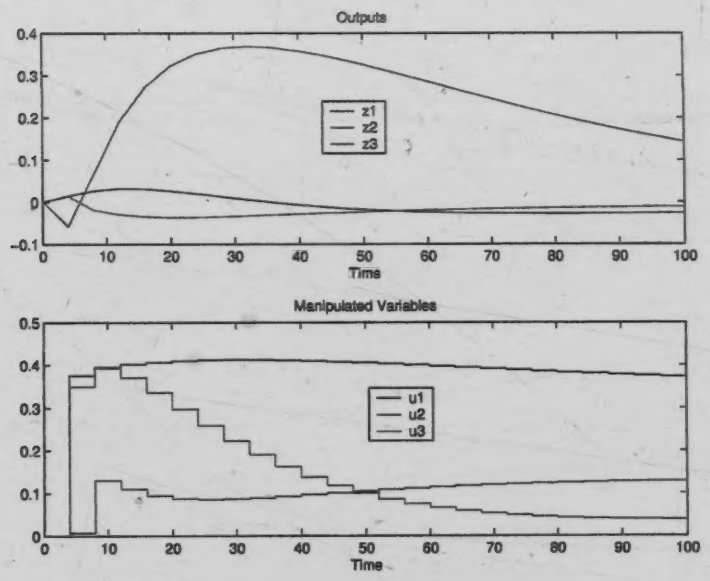


(a) Using $Q = diag(2,\ 2,\ 0.1,\ 0.0001)$ and $R = diag(0.1,\ 0.1,\ 0.1)$

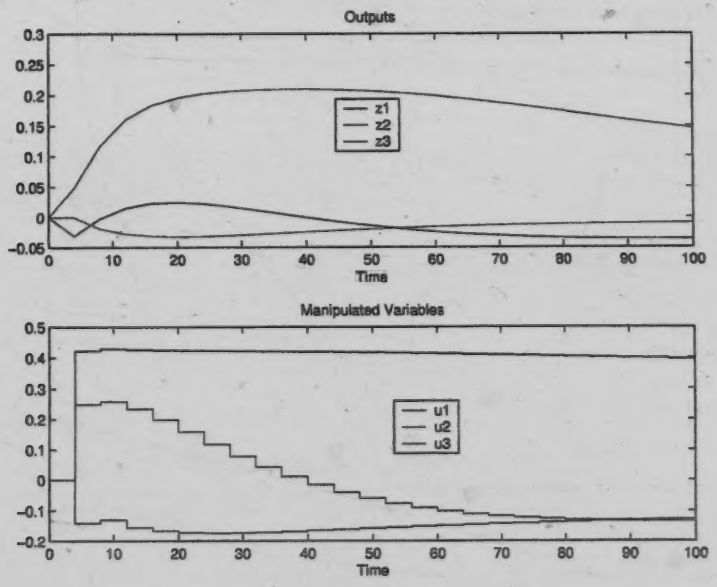


(b) Using $Q = diag(2,\ 4,\ 0.2,\ 0.001)$ and $R = diag(0.6,\ 0.3,\ 0.3)$

Figure 2.8: Closed-Loop Response of Case 1

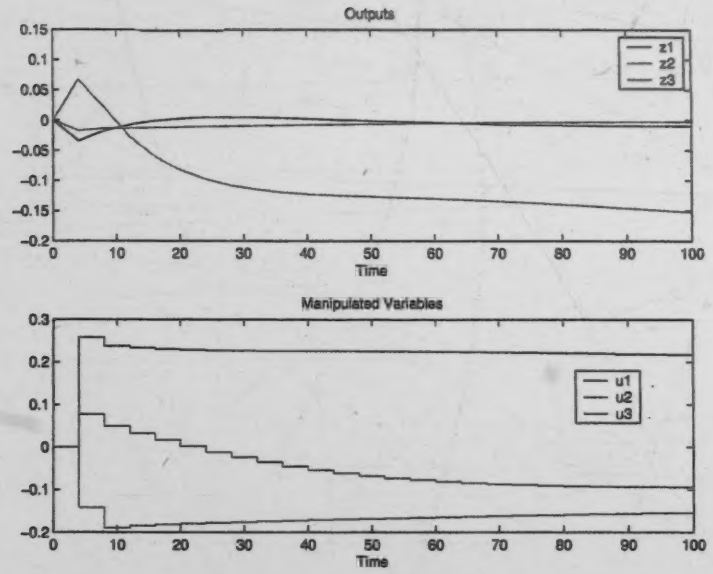


(a) Closed-Loop Response for Case II

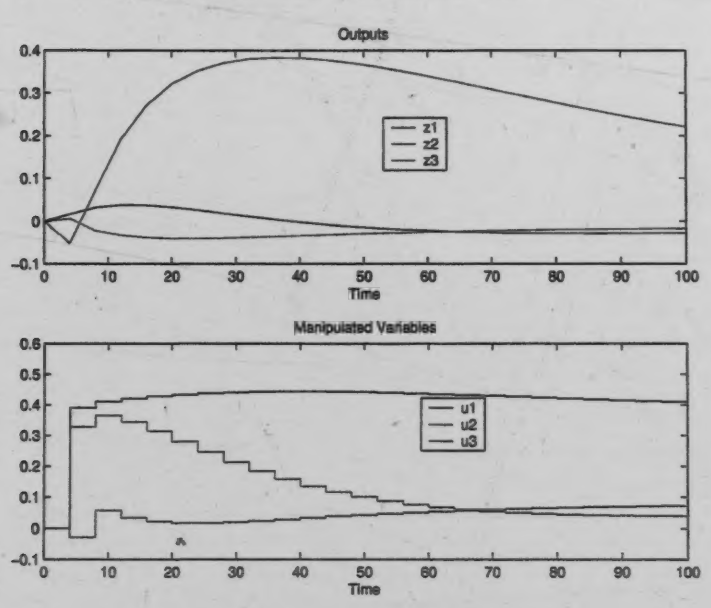


(b) Closed-Loop Response for Case III

Figure 2.9: Simulation Results for Case II and III Using Active Set Method



(a) Closed-Loop Response for Case IV



(b) Closed-Loop Response for Case V

Figure 2.10: Simulation Results for Case IV and V Using Active Set Method

Table 2.9: Numerical Results for the Full space Method

Problem	n	nbnb	Backslash			Iterative		
			nfn	CPU(s)	iter	nfn	CPU(s)	iter
Case 1	260	0	37	0.5044	2	42	0.5993	2
Case 2	260	0	37	0.5336	2	42	0.6201	2
Case 3	260	0	37	0.4799	2	42	0.5908	2
Case 4	260	0	37	0.4699	2	42	0.6442	13
Case 5	260	0	37	0.4852	2	42	0.5412	2

2.9 Conclusion

The main conclusions on active set methods implementations to large scale MPC problem are the followings:

- Active set methods are best suited for MPC problem with prior information about the optimal active constraints or for MPC applications in which the active constraints rarely change during the iteration. This is because the number of iterations required to solve the QP problem depends on accurately identifying the active constraints.

- A feasible starting point must exist in order to implement active set method on MPC problem; thus, we may need to soften the constraints or set an output constraint window to ensure feasibility of the MPC problem.

- Null space active set method is efficient for MPC applications with large number of constraints compared to the number of decision variables because the optimization steps are performed in the space containing only the manipulated variables.

- Full-space (direct inversion and Krylov subspace iterative) active set methods take advantage of the MPC problem structure and the sparsity of the model during the optimization steps.

- Implementation of Krylov subspace iterative techniques improve the applicability of active set methods to large scale MPC problems. As presented in Chapter 5, the implementation of Krylov subspace iterative method to active set method reduces the cost of solving the tested QP from the prototypical $O(N^3)$ to $O(N^{\frac{1}{2}})$.

- All the improved active set methods solve the MPC case studies problem accurately with no constraint violation as shown in Figure 2.4 and Figure 2.8 through 2.10.

- Table 2.8 and 2.9 confirm that Krylov subspace iterative method effectively exploit sparseness and require less CPU time when compared to matrix factorization methods.

Chapter 3

INTERIOR POINT METHODS

3.1 Introduction

Interior point methods are more efficient for MPC applications if the active set changes more rapidly when the system is subject to high disturbance effect or frequent set-point changes. This is because the number of iterations required to solve the QP is independent on the number of active constraints. Interior point methods incorporate both the active and inactive constraints associated with the MPC problem into the QP solution at every iteration.

Unlike active-set methods which proceed along the boundary of the feasible region, interior point methods stay strictly within the interior of the feasible region by following a path within the feasible region. Interior point algorithms usually require solving systems of nonlinear algebraic equation using improved Newton's method. Different path following methods are described in Section 3.2 and 3.3 with more emphasis on Mehrotra Predictor-Corrector Method for MPC applications.

In general, interior point methods for convex quadratic programs usually provide algorithms that are simple to describe, easy to implement, and efficient for large, sparse systems. They approach the minimum of a convex QP given in Eq 1.9 by approximating the solution of the optimality conditions provided in Eq 1.11 using Newton's method. The Newton algorithm finds the step direction at each iteration

by solving

$$\nabla F(x_k, \pi_k, \lambda_k, t_k) \begin{bmatrix} \Delta x_k \\ \Delta \pi_k \\ \Delta \lambda_k \\ \Delta t_k \end{bmatrix} = -F(x_k, \pi_k, \lambda_k, t_k)$$

where ∇F is the Jacobian matrix of the KKT optimality condition. The Newton's method of solving system of nonlinear algebraic equation is reviewed in Section A.3. The Newton's method could violate constraints when ∇F becomes ill-conditioned as the solution is being approached (Wright 1992). In order to ensure feasibility, a line search is required to know the desired step length so as not to violate any constraints. Thus, the next iterate is obtained as

$$\begin{bmatrix} x_{k+1} \\ \pi_{k+1} \\ \lambda_{k+1} \\ t_{k+1} \end{bmatrix} = \begin{bmatrix} x_k \\ \pi_k \\ \lambda_k \\ t_k \end{bmatrix} + \alpha_k \begin{bmatrix} \Delta x_k \\ \Delta \pi_k \\ \Delta \lambda_k \\ \Delta t_k \end{bmatrix}$$

where α_k is the step-length. The iteration is repeated until convergence. Steps of a prototypical interior point method are provided in Algorithm 2.

3.2 Path Following Method

A path following method as described by Wright (1997*c*) and Mehrotra (1992*b*) defines a path of iterates which ensures strict feasibility of each iterate. The method augments the convex QP problem by introducing logarithmic barrier functions (Wright 1997*c*).

$$\begin{aligned} \min_x q(x) &= \frac{1}{2} x^T G x + d^T x - \tau \sum_{i=1}^{m} \log(t_i) \\ s.t & \\ Fx - g &= 0 \\ Cx - h + t &= 0 \end{aligned} \tag{3.1}$$

Where t_i is the slack variable associated with the i-th inequality constraint and τ is the path shaping parameter required to keep the iterates strictly feasible. Forming

the Lagrangian we have,

$$L(x,\pi,\lambda,t) = \frac{1}{2}x^T Gx + d^T x - \tau \sum_{i=1}^{m} \log t_i + \pi^T(Fx-g) + \lambda^T(Cx-h+t) \quad (3.2)$$

The first order optimality conditions become

$$\begin{aligned} \nabla_x L &= Gx + d + F^T\pi + C^T\lambda = 0 \\ \nabla_t L &= -\tau\, T^{-1}e + \lambda = 0 \\ \nabla_\pi L &= Fx - g = 0 \\ \nabla_\lambda L &= Cx - h + t = 0 \end{aligned} \quad (3.3)$$

Rewriting the system of equations in Eq 3.3, the optimality conditions become

$$\begin{aligned} Gx + d + F^T\pi + C^T\lambda &= 0 \\ Fx - g &= 0 \\ Cx - h + t &= 0 \\ T\Lambda e &= \tau e \end{aligned} \quad (3.4)$$

Therefore, the minimizers x_τ of the logarithmic barrier subproblem in Eq 3.1 are the x components of the central path vectors $(x_\tau, \pi_\tau, \lambda_\tau, t_\tau) \in \mathcal{C}$. As the iteration proceeds τ approaches zero and x_τ approaches some solution x^* of the convex QP program (Wright 1997*c*). The central path approach tends to be slow and may be generalized by adding a centering parameter $\sigma \in [0,1]$ and duality measure, μ:

$$\mu = \frac{1}{m}\sum_{i=1}^{m} \lambda_i t_i = \frac{\lambda^T t}{m}$$

The choice of centering parameter varies from method to method. The general framework for the path following primal-dual interior point method is given in Algorithm (4).

The different path following methods differ only in the way the step direction is modified and the centering parameter is calculated. The most successful path following method is the Mehrotra Predictor-Corrector method.

Start at some feasible point$(x_0, \lambda_0, \pi_0, t_0)$

Set $\sigma_k \in [0,1]$ so that $(\lambda_{k+1}, t_{k+1}) > 0$, $\mu_k = \frac{\lambda_k^T t_k}{m}$

Calculate Newton's step

$$\begin{bmatrix} G & -F^T & -C^T & 0 \\ -F & 0 & 0 & 0 \\ -C & 0 & 0 & -I \\ 0 & 0 & T & \Lambda \end{bmatrix} \begin{bmatrix} \Delta x_k \\ \Delta \pi_k \\ \Delta \lambda_k \\ \Delta t_k \end{bmatrix} = \begin{bmatrix} -r_G \\ -r_F \\ -r_C \\ -T_k \Lambda_k e + \sigma_k \mu_k e \end{bmatrix} \tag{3.5}$$

Line search for α_k and set

$$(x_{k+1}, \pi_{k+1}, \lambda_{k+1}, t_{k+1}) = (x_k, \pi_k, \lambda_k, t_k) + \alpha_k(\Delta x_k, \Delta \pi_k, \Delta \lambda_k, \Delta t_k)$$

Repeat the steps until convergence

Algorithm 4: Path Following Method Framework

3.3 Mehrotra Predictor-Corrector Method

Most interior point codes are based on the Mehrotra Predictor-Corrector (MEPC) algorithm described in Mehrotra (1992*b*) and Mehrotra (1992*a*). The MEPC algorithm differs from the basic primal-dual interior point algorithm in that it introduces a correction step that enhances centrality and satisfaction of the positivity constraints $(\lambda, t) \geq 0$ for the basic Newton step (predictor step).

The search direction at each iteration is calculated from three components (Wright 1997*c*)

- *An affine-scaling or predictor direction* - This is a pure Newton direction for the function $F(x, \lambda, \pi, t)$ defined by

$$\begin{bmatrix} G & -F^T & -C^T & 0 \\ -F & 0 & 0 & 0 \\ -C & 0 & 0 & -I \\ 0 & 0 & T & \Lambda \end{bmatrix} \begin{bmatrix} \Delta x^{\text{aff}} \\ \Delta \pi^{\text{aff}} \\ \Delta \lambda^{\text{aff}} \\ \Delta t^{\text{aff}} \end{bmatrix} = - \begin{bmatrix} r_G \\ r_F \\ r_C \\ \Lambda T e \end{bmatrix} \tag{3.6}$$

where $r_G = Gx + d - F^T - C^T\lambda$, $r_F = -Fx + g$, $r_C = -Cx - t + h$. This step focusses on reducing the duality measure μ.

- *A centering term*, $\sigma \in (0,1)$. When $\sigma_k = 0$ the search direction is the pure Newton step for nonlinear systems of algebraic equations and the resulting method is an affine scaling algorithm. The effect of choosing positive values of σ_k is to orient the step away from the boundary of the nonnegative orthant defined by $(\lambda, t) \geq 0$.

- *A corrector direction*- The corrector direction focuses on improving centrality by attempting to bring the iterates back to the central path

$$\begin{bmatrix} G & -F^T & -C^T & 0 \\ -F & 0 & 0 & 0 \\ -C & 0 & 0 & -I \\ 0 & 0 & T & \Lambda \end{bmatrix} \begin{bmatrix} \Delta x^{cc} \\ \Delta \pi^{cc} \\ \Delta \lambda^{cc} \\ \Delta t^{cc} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ -\Delta\Lambda^{aff}\Delta T^{aff} + \sigma\mu e \end{bmatrix} \tag{3.7}$$

The MEPC algorithm is shown in Algorithm (5).

3.4 Overview of IP Approaches for QP

The interior point solver **LOQO** invented by Vanderbei (1998) and the Riccati recursion IP solver by Rao *et al.* (1998) are used for this review. As described by Bartlett *et al.* (2002), each method represents a different interior point approach of solving QP from MPC.

3.4.1 LOQO

The algorithm implemented in **LOQO** (Vanderbei 1998) is a primal-dual interior point method. It uses a special sparse symmetric quasi-definite linear solver to solve the required IP linear algebra. The main computational load in **LOQO** is the formulation and factorization of the sparse symmetric quasi-definite system below.

$$\begin{bmatrix} -(H+D) & A^T \\ A & E \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} b \\ c \end{bmatrix} \tag{3.8}$$

In which, D and E are diagonal matrices. A represents the Jacobian of all the general equality and inequality constraints and H is the Hessian matrix. The results

Choose tolerance ϵ and starting point x, π, λ, t with $t > 0$, $\lambda > 0$

1. Evaluate stopping criteria, $\|\mu_k\| \leq \epsilon$ and $\|r_G\| \leq \epsilon$
2. Compute affine-scaling direction by solving Eq 3.6
3. Compute steps to the boundary

$$\begin{aligned} \alpha_x &= \max\{\alpha \in [0,1] | t + \alpha \Delta t^{\text{aff}} \geq 0\} \\ \alpha_\lambda &= \max\{\alpha \in [0,1] | \lambda + \alpha \Delta \lambda^{\text{aff}} \geq 0\} \end{aligned}$$

4. Compute centering-corrector steps by solving Eq 3.7 with

$$\sigma = \left(\frac{(t + \alpha_x \Delta t^{\text{aff}})^T (\lambda + \alpha_\lambda \Delta \lambda^{\text{aff}})}{t^T \lambda} \right)^3$$

5. Compute search direction

$$\begin{aligned} \Delta x &= \Delta x^{\text{aff}} + \Delta x^{\text{cc}} \\ \Delta \pi &= \Delta \pi^{\text{aff}} + \Delta \pi^{\text{cc}} \\ \Delta \lambda &= \Delta \lambda^{\text{aff}} + \Delta \lambda^{\text{cc}} \\ \Delta t &= \Delta t^{\text{aff}} + \Delta t^{\text{cc}} \end{aligned}$$

6. Determine step sizes and update

$$\begin{aligned} \alpha_x &= \max\{\alpha > 0 | t + \alpha \Delta t \geq 0\} \\ \alpha_\lambda &= \max\{\alpha \geq 0 | \lambda + \alpha \Delta \lambda \geq 0\} \end{aligned}$$

$$\begin{aligned} x &= x + \min\{1, 0.99\alpha_x\} \Delta x \\ \pi &= \pi + \min\{1, 0.99\alpha_x\} \Delta \pi \\ \lambda &= \lambda + \min\{1, 0.99\alpha_\lambda\} \Delta \lambda \\ t &= t + \min\{1, 0.99\alpha_x\} \Delta t \end{aligned}$$

Repeat step $1 \rightarrow 6$ until convergence *i.e* $\|\mu_k\| \leq \epsilon$ and $\|r_G\| \leq \epsilon$

Algorithm 5: Mehrotra's Predictor-Corrector Method

in Bartlett *et al.* (2002) showed that the cost associated with this matrix is very significant and does not scale very well with the problem size.

3.4.2 Riccati Recursion with IP

Wright (1993) and Rao *et al.* (1998) have studied quadratic programming formulations of optimal control problems with linear constraints. They showed that the Riccati recursion of (unconstrained) linear-quadratic optimal control can be used to compute the search directions in an interior point method very fast, *i.e.*, at a cost that is linear in the prediction horizon $O(N)$ and cubic in the system dimensions. The computational cost of the entire process was described by Rao *et al.* (1998) as $O(N(n_x + n_u)^3)$ where n_x and n_u denote the number of component of the state and input vectors respectively .

3.4.3 IP method Code developed for this work

Interiormpc

Like the QP algorithm by Rao *et al.* (1998) and **ISQP** by Albuquerque *et al.* (1999), the proposed IP method implements the MEPC algorithm described in Section 3.3. However, there are number of advances that have been made to this approach for efficient applications to QP problem from MPC.

Through **Interiormpc**, an augmented system containing only the primal variables and the equality multipliers is used for the Newton's step. This formulation results to a special class of sparse symmetric matrix described in Section 3.5.2. The reduced systems in the predictor and corrector steps are then solved using the Krylov subspace iterative method (Preconditioned Bi-CGSTAB) described in Chapter 2. This approach exploits the sparse matrix structure of the reduced KKT systems and not destroy it as in the recursive recursion approach. The incomplete LU factors in the prediction step are also used as preconditioners in the corrector step, thus reducing the additional cost of solving two systems of equations. Also hot starting of the primal variables and cold starting of the slacks and inequality multipliers are implemented so as to reduce the iterative step for efficient computation. The cost of computation increases linearly with the prediction horizon, $O(N)$. This agrees with the result

obtained by Rao *et al.* (1998). Typical of IP algorithms, the number of iterations required to solve the tested quadratic programming problems is independent on the number of the inequalities in the problem. The simulation results in Chapter 5 reveals that approximately ten interior point iterations are required by QP.

3.5 Improvements on the Prototypical IP Method

The improvement on the basic IP algorithms involves hot starting for primal variables and warm starting for dual and slack variables. Our improvements reduce the iteration steps and exploit the sparsity of the KKT equations through the implementation of the Krylov subspace iterative method to an augmented system.

3.5.1 Initial Point

A good initial point usually reduces the number of iteration required to get to the optimum value (Wright 1997*c*). Using the least squares approach described in Chapter 2, a good starting point of the primal variable x is obtained. Choosing large values for the slacks t and the dual variables λ at the beginning of the iteration prevents ill-conditioning of the Jacobian matrix for smooth convergence of the Newton's iteration (Mehrotra 1992*a*). For MPC application, the primal, dual and slacks variables of the previous open loop-minimization are updated and used as the starting point for the next minimization.

3.5.2 Augmented System

The major computational operation in the IP algorithm is the Newton step (step direction calculation). Primal-dual Interior point method introduces more variables into the optimization problem and this increases the size of the Jacobian matrix of the Newton equations. For this work, the KKT equations in Eq 3.5 is solved using an augmented system with a special class of symmetric coefficient matrix. Augmented system formulation speeds up the convergence of the Newton's step and reduces the size of the Jacobian matrix from $\mathbb{R}^{(N+m_e+2m_i)\times(N+m_e+2m_i)}$ to $\mathbb{R}^{(N+m_e)\times(N+m_e)}$. The augmented matrix is obtained by substituting the following set of equations for Δt

and $\Delta\lambda$ respectively in the Newton Equations.

$$\begin{aligned} \Delta t &= -\Lambda^{-1}(r_t + T\Delta\lambda) \\ \Delta\lambda &= -\Lambda T^{-1}(C\Delta x + \Lambda^{-1} r_T - r_C) \end{aligned} \tag{3.9}$$

Because Λ and T are diagonal matrices containing the dual and slack variables respectively their inverse computation is easy. Therefore, the augmented system of the Newton equations become

$$\begin{bmatrix} (G + C^T \Lambda T^{-1} C) & -F^T \\ -F & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta\pi \end{bmatrix} = \begin{bmatrix} -r_G - C^T T^{-1}(r_T - \Lambda r_C) \\ -r_F \end{bmatrix} \tag{3.10}$$

in

$$\begin{aligned} r_T^{\text{aff}} &= T\Lambda e \\ r_T^{\text{cc}} &= \Delta T^{\text{aff}} \Delta\Lambda^{\text{aff}} - \sigma\mu e \end{aligned}$$

The coefficient matrix in Eq 3.10 is of an important class of symmetric matrix in which $(G + C^T \Lambda T^{-1} C)$ is symmetric positive definite and F^T has a full column rank. The matrices, G, C, and F are constants in the IP method, unlike in the active set method. The preconditioned Bi-CGSTAB iterative method is used to solve this linear system efficiently with the preconditioners obtained from the incomplete LU factorization of the augmented matrix in Eq 3.10.

3.6 Application of IP Method to Case Studies

The developed IP algorithm is implemented on the case studies described in Chapter 2. Identical simulation results show that interior point method and active set methods both converge to the same result; however, the computational load is $O(N)$ for IP, $O(N^3)$ for the reduced space active set method and $O(N^{\frac{1}{2}})$ for active set method with Krylov subspace techniques implementations.

3.6.1 Two Tank System

The following sets of equations are obtained using prediction horizon $N = 5$ and control horizon $H_u = 3$ for the Two Tank problem described in Chapter 2.

Cost Function

$$\min_X \frac{1}{2} X^T G X + D^T X$$

in which

$$G = \begin{bmatrix} Q & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & R & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & Q & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & R & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & Q & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & R & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & Q & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \overline{Q} \end{bmatrix} \quad D = -2 \begin{bmatrix} Qx_s \\ Ru_s \\ Qx_s \\ Ru_s \\ Qx_s \\ Ru_s \\ Qx_s \\ \overline{Q}x_s \end{bmatrix} \quad X = \begin{bmatrix} x(k+1) \\ u(k) \\ x(k+2) \\ u(k+1) \\ x(k+3) \\ u(k+2) \\ x(k+4) \\ x(k+5) \end{bmatrix}$$

Equality Constraints

The discrete time state space model equation yields

$$\begin{bmatrix} -I & B & 0 & 0 & 0 & 0 & 0 & 0 \\ A & 0 & -I & B & 0 & 0 & 0 & 0 \\ 0 & 0 & A & 0 & -I & B & 0 & 0 \\ 0 & 0 & 0 & 0 & A & B & -I & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & A & -I \end{bmatrix} X = \begin{bmatrix} -Ax(k) \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Inequality Constraints

The followings inequality constraints are obtained as a results of the hard bounds constraints on the manipulated variable.

$$\begin{bmatrix} -2 \\ -2 \\ -2 \end{bmatrix} \leq \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} X \leq \begin{bmatrix} 2 \\ 2 \\ 2 \end{bmatrix}$$

With weighting matrices $Q = diag[2, 2]$ and $R = 0.1$, the resulting closed loop simulation results are shown in Figure 3.1. This is identical to the result obtained

using active set method. The process states settle to the required steady state values after $0.4s$ without any input constraint violation. The computational results of this applications are presented and discussed in detail in Chapter 5.

3.6.2 Shell Heavy Oil Fractionator

Figures 3.2, 3.3 and 3.4 show the closed-loop simulation results obtained from the application of the improved IP method to the Shell Heavy Oil Fractionator case study described in Chapter 2. The result obtained is the same as the result obtained using active set method.

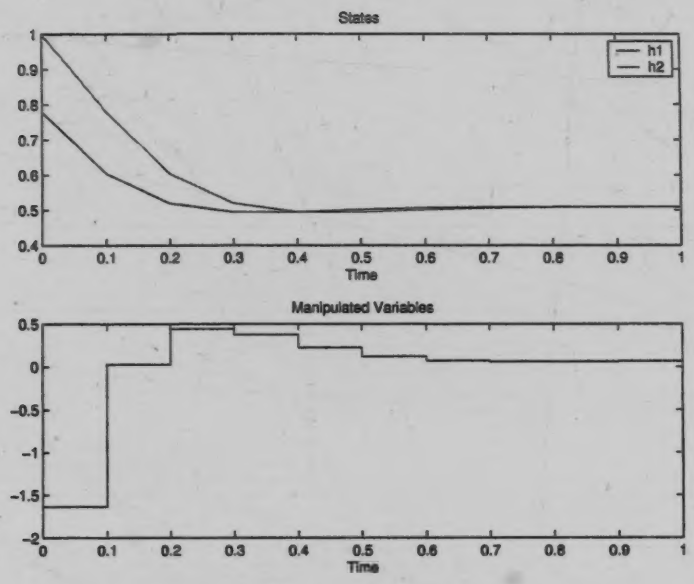
Table 3.1 gives the numerical results of implementing the Krylov subspace IP method (**Activesetconju**) to solve the QP problems from all the prototype test cases presented in Table 2.7. Comparing Table 3.1 to Table 2.8 and 2.9, it is shown that the proposed IP method requires more iterations than active set methods but less CPU time when compared with reduced space active set methods in Table 2.8.

Table 3.1: Numerical Results for the Interior Point Method

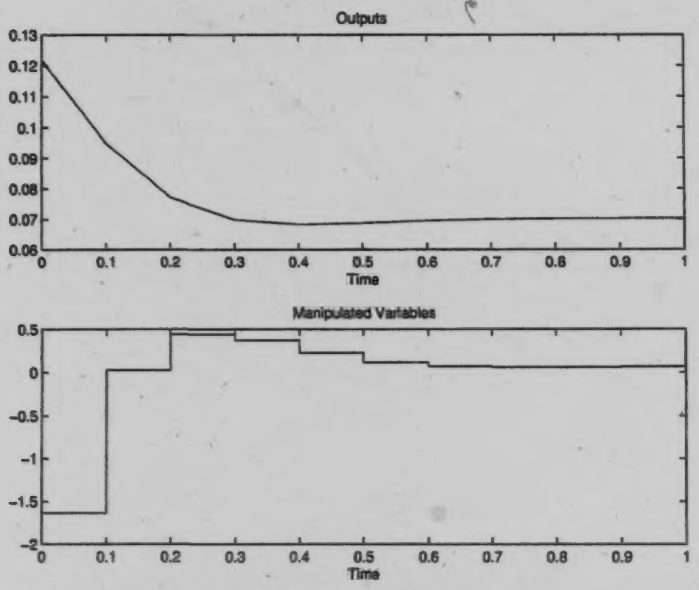
	Case I	Case II	Case III	Case IV	Case V
CPU(s)	0.6201	0.5924	0.5464	0.4795	0.4728
$\mathcal{I}(x^*)$	1.3171	2.7103	2.0209	-0.9601	3.0326
nbnb	0	0	0	0	0
nfn	23	23	23	22	23
n	260	260	260	260	260
iter	7	4	4	3	4

3.7 Conclusions

The main conclusions of this chapter include:

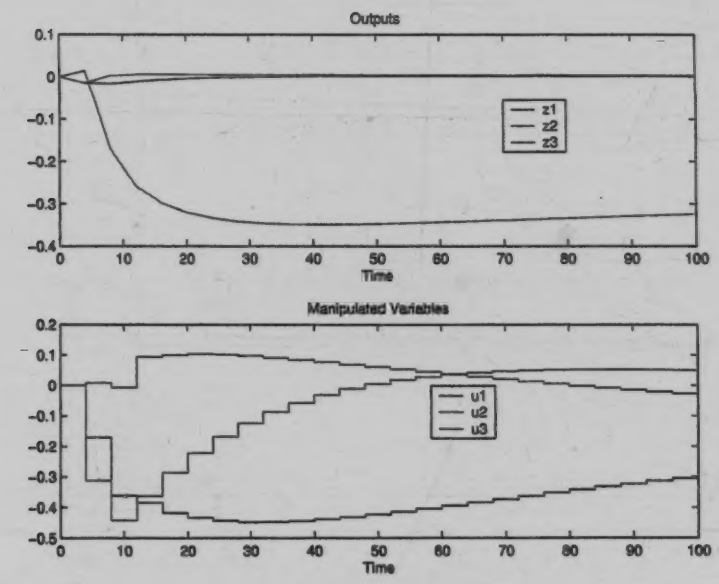


(a) Closed-Loop MPC Response for States and Input

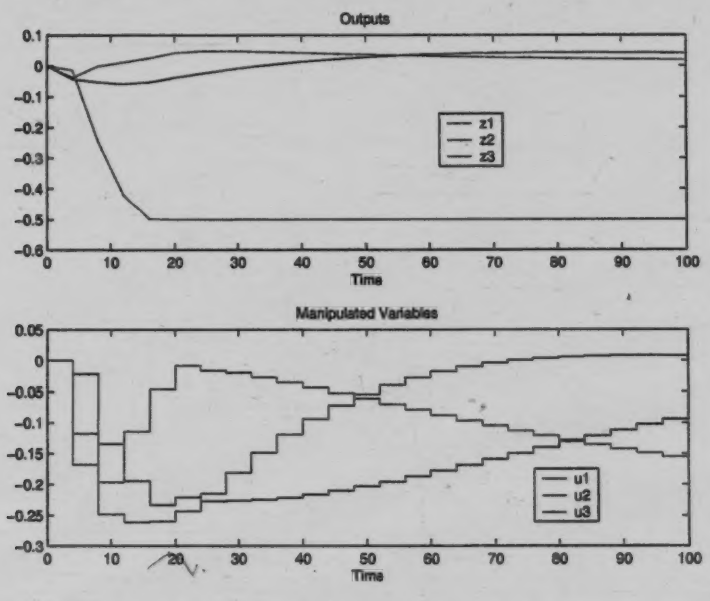


(b) Closed-Loop MPC Response for Output and Input

Figure 3.1: Simulation Results of the Two tank System Using Interior Point Method

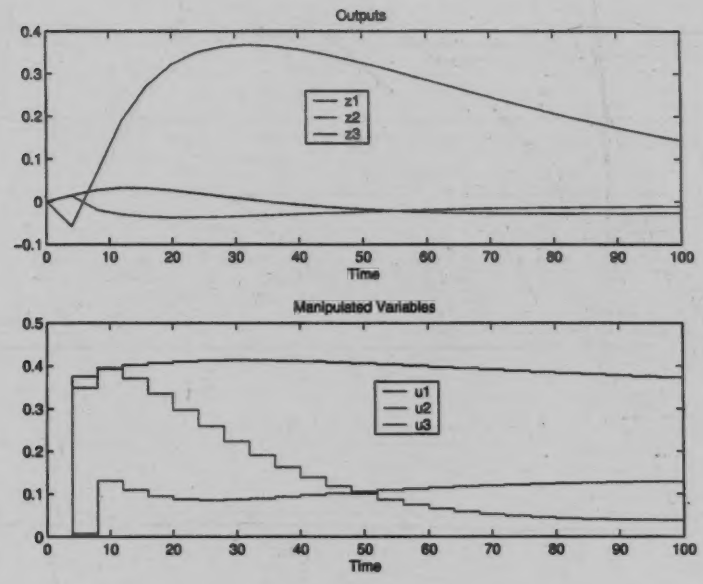


(a) Using $Q = diag(2,\ 2,\ 0.1,\ 0.0001)$ and $R = diag(0.1,\ 0.1,\ 0.1)$

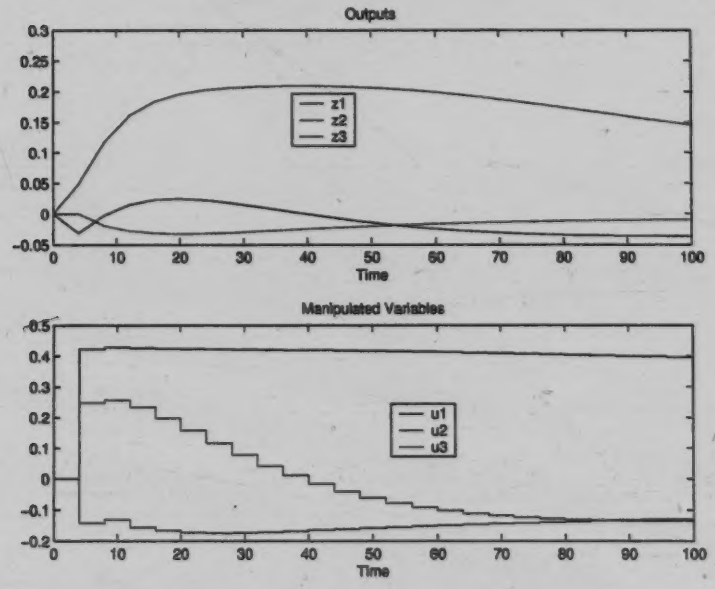


(b) Using $Q = diag(2,\ 4,\ 0.2,\ 0.001)$ and $R = diag(0.6,\ 0.3,\ 0.3)$

Figure 3.2: Closed-Loop Response of Case 1

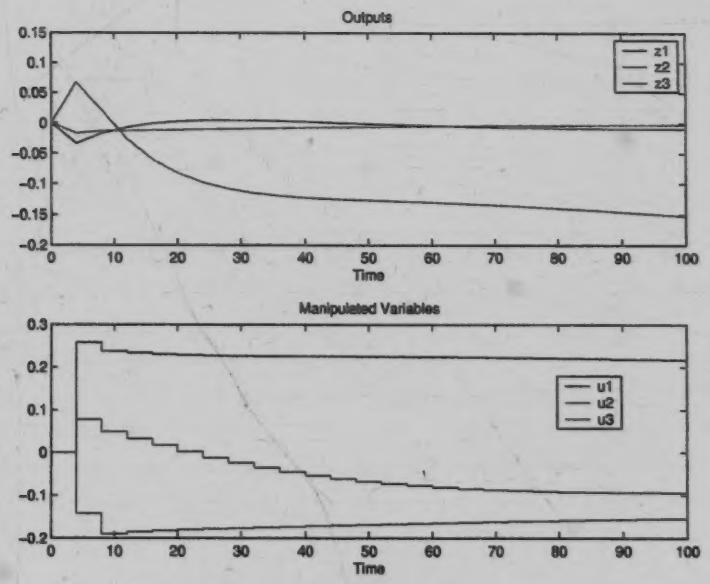


(a) Closed-Loop Response for Case II

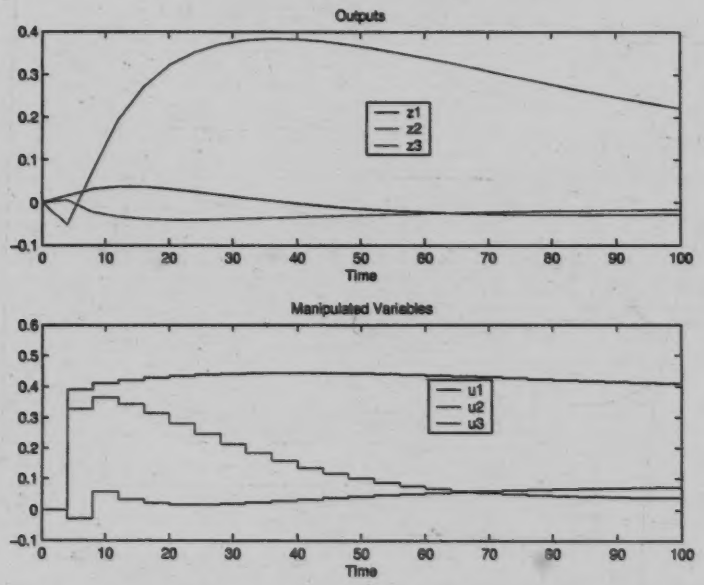


(b) Closed-Loop Response for Case III

Figure 3.3: Simulation Results for Case II and III Using Interior Point Method



(a) Closed-Loop Response for Case IV



(b) Closed-Loop Response for Case V

Figure 3.4: Simulation Results for Case IV and V Using Interior Point Method

1. Interior point methods are more efficient than active set methods if the active set changes frequently as a result of high disturbance effect or frequent set-point changes. This is because the number of iteration required to solve QP is independent on the number of active constraints.

2. The Primal-Dual IP method takes advantage of the MPC optimization problem structure and the sparsity of the model during the computation of the optimization step. Through the implementations of the Krylov subspace iterative techniques for the linear algebra.

3. From the tested simulations, Methrotra's Predictor-Corrector method sometimes requires more iterations than active set methods but these steps are relatively inexpensive for large sparse matrices.

4. Based on our findings cold starting (choose large initial values) of the slack and dual variables at the onset of the iteration prevents ill-conditioning of the Jacobian matrix for smooth convergence of the Newton's step.

5. Table 2.8 and Table 3.1 show that the proposed IP method requires less CPU time when compared with reduced space active set methods for large system.

6. The simulation results on Figures 2.8 – 2.10 are identical to those on Figures 3.2 – 3.4 thus supporting the uniqueness of the optimization solution.

Chapter 4

MATLAB OPTIMIZATION TOOLBOX

4.1 Introduction

This chapter focuses on the application of the *Matlab* Optimization Toolbox to MPC problems. Also, it identifies the major differences and similarities between *Matlab* application results and those obtained from the proposed optimization algorithms in Chapter 2 and 3.

The Optimization Toolbox provides the *Matlab* user with tools for both general and large scale optimization of linear, quadratic, nonlinear least squares, and general nonlinear programming. The Optimization Toolbox algorithms include standard optimization techniques for solving large sparse problems. The principal algorithms for constrained minimization are the Sequential Quadratic Programming (SQP) and the Interior-Reflective Newton method which is described in (Coleman and Li 1996). The two methods are used for constrained MPC problem in which constraints can either be equality or inequality, *Matlab* quadratic programming routine known as `quadprog` is mainly for QP problem from MPC. The `quadprog` provides algorithms for large-scale systems with quadratic objective and linear bounds or linear equalities. The medium-scale algorithm in `quadprog` is used for QP problems with linear inequalities or both linear equalities and bounds constraints.

4.1.1 Large-Scale Algorithm

Large-scale algorithm is designed for quadratic programming problems with only equality or bounds constraints and not both.

Linear Equality Constrained

The general large scale equality constrained QP problem can be written as

$$\min_x \{ 0.5x^T Hx + f^T x \text{ such that } Ax = b\} \tag{4.1}$$

where A is an m-by-n matrix ($m \leq n$). In order to solve the problem in Eq 4.1, the **quadprog** preprocesses the Jacobian matrix A to remove strict linear dependencies using a technique based on the LU-factorization of A^T (Mathworks 2002) and computes an initial feasible point x_0 using a sparse least squares step and proceeds with the step direction calculation by solving a QP subproblem via Preconditioned Conjugate Gradient (PCG) algorithm. The basic linear algebra step involves systems of the form

$$\begin{bmatrix} C & \tilde{A}^T \\ \tilde{A} & 0 \end{bmatrix} = \begin{bmatrix} r \\ 0 \end{bmatrix} \tag{4.2}$$

in which $\tilde{A}$ approximates A, and C is a sparse symmetric positive-definite approximation of the Hessian (Mathworks 2002).

Box Constraints

The box constrained problem is of the form

$$\min_x \{ 0.5x^T Hx + f^T x \text{ such that } l \leq x \leq u\} \tag{4.3}$$

where l is a vector of lower bounds and u is a vector of the upper bounds on the decision variable vector x. The Optimization Toolbox solves problem 4.3 through an interior-reflective Newton method.

4.1.2 Medium-Scale Algorithm

For QP problems with linear inequalities or both linear equalities and bounds constraints, the Optimization Toolbox uses medium scale algorithm. This algorithm finds

an initial feasible point in the center of the bounds by solving a linear programming problem and solves the QP problem using null space active-set method implementing either Newton or steepest descent algorithms for step direction calculation and proceeds with a line search to determine the step length parameter α_k that sufficient decrease the cost function, it then update and repeat the QP subproblem calculation until convergence.

4.2 Case Studies

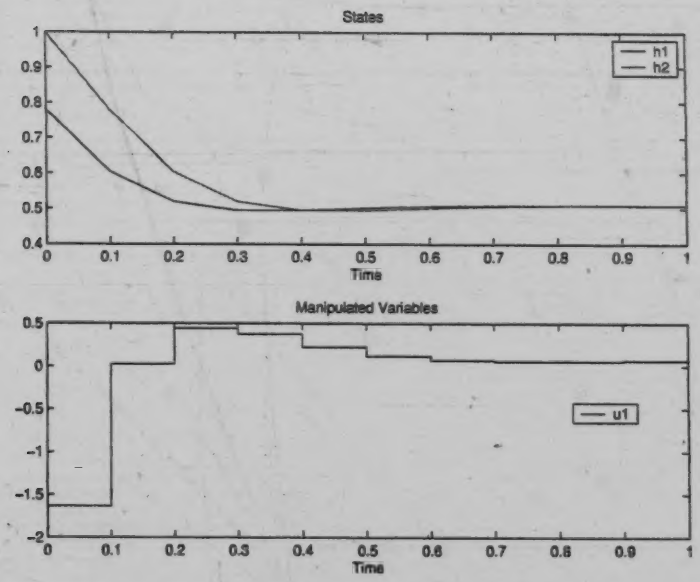
MPC problems usually have both equality and inequality constraints and thus the **quadprog** selects medium-scale algorithm of solution because the large-scale algorithm is strictly for equality or bounds constrained QP problem.

4.2.1 Two Tank System

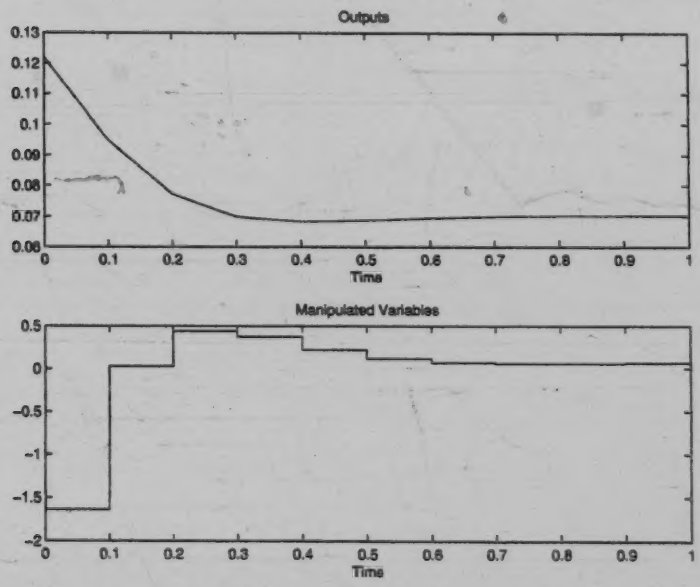
Figure 4.1 shows the obtained simulation results for the Two Tank System presented in Chapter 2 using the *Matlab* **quadprog** for the optimization. The simulation results are identical to the previous results from the developed optimization algorithms.

4.2.2 Shell Heavy Oil Fractionator Column

Table 4.1 gives the summary of the numerical results of *Matlab* **quadprog** application to Shell Heavy Oil Fractionator Column protype test cases described in Chapter 2 and Figure 4.2 – 4.4 show the closed loop simulation results. The CPU time in Table 4.1 are similar to those obtained using the reduced space active set method shown in Table 2.8, thus, supporting *Matlab* medium scale algorithm implementations for MPC applications. The closed loop simulation results are identical to the obtained results from the developed active set and interior point algorithms.

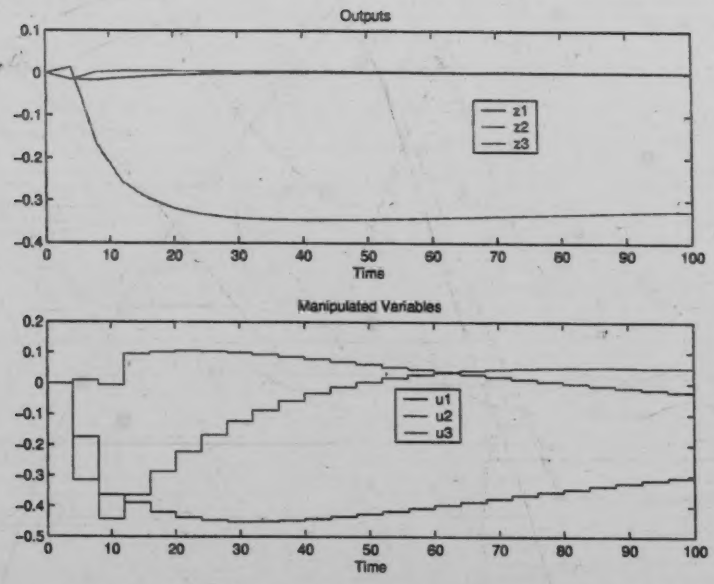


(a) Closed-Loop MPC Response for States and Input

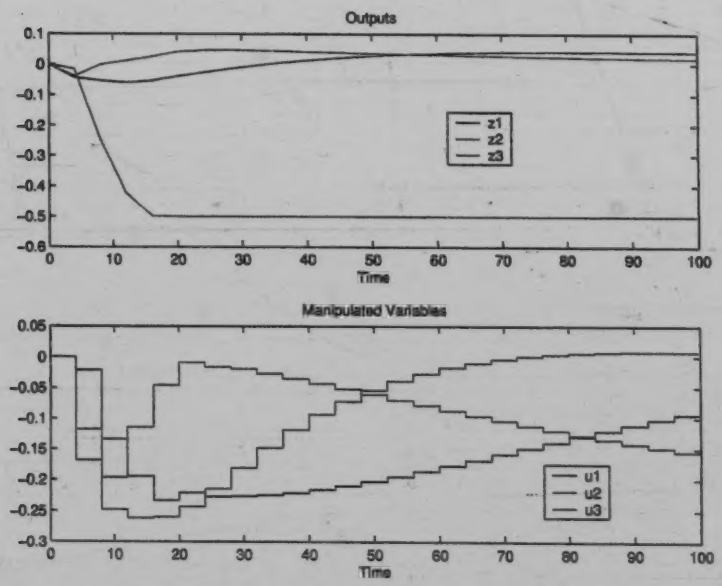


(b) Closed-Loop MPC Response for Output and Input

Figure 4.1: Simulation Results of the Two tank System Using Optimization Toolbox in *Matlab*

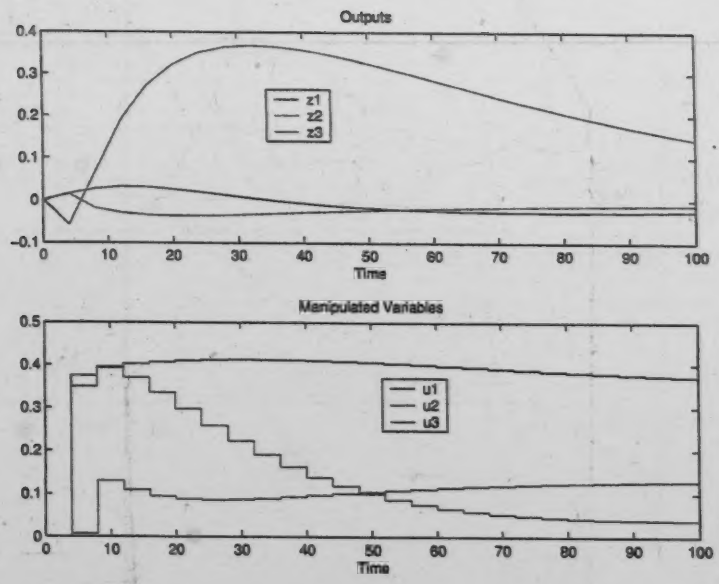


(a) Using $Q = diag(2,\ 2,\ 0.1,\ 0.0001)$ and $R = diag(0.1,\ 0.1,\ 0.1)$

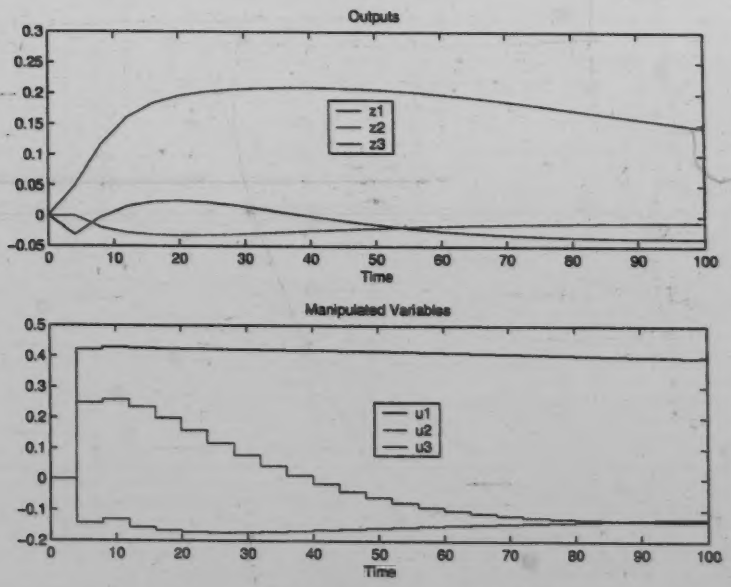


(b) Using $Q = diag(2,\ 4,\ 0.2,\ 0.001)$ and $R = diag(0.6,\ 0.3,\ 0.3)$

Figure 4.2: Closed-Loop Response of Case 1

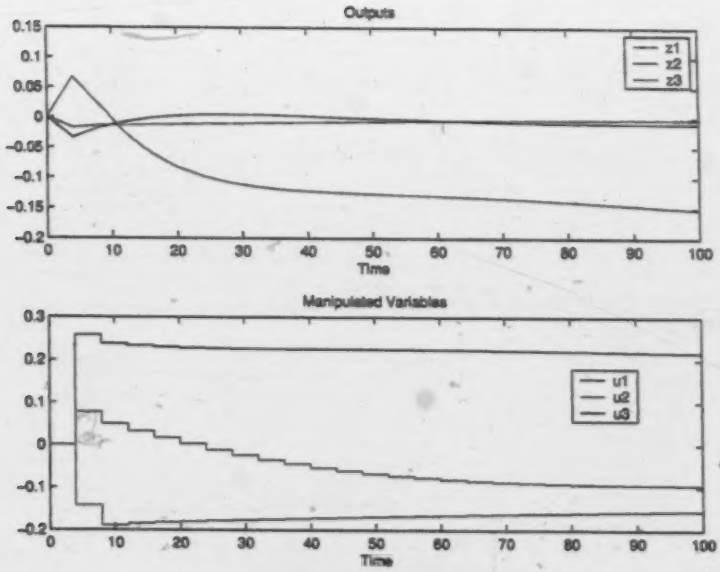


(a) Closed-Loop Response for Case II

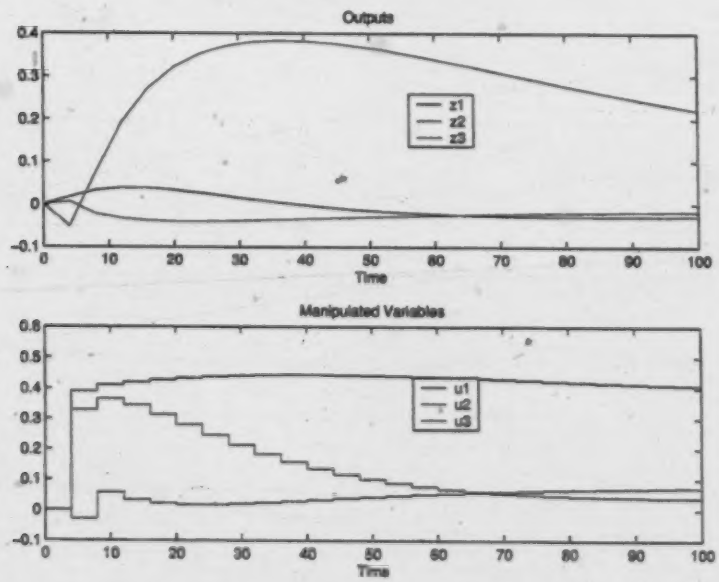


(b) Closed-Loop Response for Case III

Figure 4.3: Simulation Results for Case II and III Using Optimization Toolbox in *Matlab*



(a) Closed-Loop Response for Case IV



(b) Closed-Loop Response for Case V

Figure 4.4: Simulation Results for Case IV and V Using Optimization Toolbox in *Matlab*

Table 4.1: Numerical Results for the Optimization Toolbox Application

	Case I	Case II	Case III	Case IV	Case V
CPU(s)	1.5026	2.1996	1.15168	1.0072	1.08312
$\mathcal{I}(x^*)$	1.3171	2.7103	2.0209	-0.9601	3.0326
nbnb	0	0	0	0	0
nfn	25	25	25	23	25
n	260	260	260	260	260
iter	3	1	1	1	1

4.3 Conclusions

The implementations of *Matlab* `quadprog` for the MPC case studies generate the following conclusions.

1. The Optimization Toolbox uses the medium-scale algorithm for large-scale MPC problems with inequality constraints.

2. The medium-scale algorithm requires fewer iterations than the primal-dual IP method but the iteration steps are relatively expensive.

3. The application requires more CPU time compared to the full space active set methods or the primal-dual IP method.

Chapter 5

COMPLEXITY STUDIES

5.1 Introduction

The detailed discussion of the different strategies to constraint softening for feasibility of the optimization problem and the results of the complexity experiments implemented in *Matlab* 6.1 on a Pentium IV-800MHz machine forms the focus of this chapter.

MPC optimization problem size depends on the dimensions n_x of states, n_y of outputs and n_u of inputs, the prediction horizon N, control horizon H_u and the number of active constraints in the open-loop optimization. The complexity experiment examines the efficacy of the developed algorithms for solving large-scale MPC problems and analyze how each algorithm scales with problem size. Complexity testing is also carried out using *Matlab* **quadprog**.

This chapter concludes with the analysis of the overall complexity results and highlights the scaling of each algorithm with MPC problem size.

5.2 Feasibility and Constraints Softening

All the developed algorithms for solving QP problems from MPC discussed in this work have been feasible point methods *i.e.*, if the initial point is feasible, all subsequent iterates are also feasible. The methods stop execution if a feasible initial point is not obtainable. Therefore, there is need to ensure feasibility of an optimization

problem before embarking on solving it.

Any point that satisfies constraints of an optimization problem is said to be feasible (Gill *et al.* 1983). The feasible region is the set of all feasible points. A problem for which there are no feasible points is termed infeasible. While convexity guarantees termination of an optimization problem, feasibility guarantees the existence of a solution to the optimization problem.

In MPC applications, input constraints, for example actuator limitations, are hard constraints and cannot be violated. Performance or safety constraints on state and output can usually be softened by allowing violation if necessary to prevent infeasibility. For example, a disturbance can push the output outside the feasible region where no allowed manipulated variable is able to bring the output back inside the feasible region at the next sampling time. Therefore, to ensure feasibility, the output constraints are relaxed or softened as

$$y_{min} - M\epsilon \leq y(t) \leq y_{max} + M\epsilon$$

where $M \in \mathbb{R}^{n_y}$ is a constant penalty vector which is related to the concern for the output constraint violation and ϵ is the amount of constraint violation (Zheng and Morari 1995). The determination of values of M and ϵ is usually by trial-and-error. Using a different approach, Muske and Rawlings set a constraint window on the output,

$$y_{min} \leq y_{k+j} \leq y_{max} \quad j = j_1, j_1 + 1 \ldots j_2.$$

The output constraints are applied from time $k+j_1$, $j_1 \geq 1$ through time $k+j_2$, $j_2 \geq j_1$. The value j_1 is chosen such that the output constraints are feasible at time k and j_2 is chosen such that feasibility of the output constraints up to time $k + j_2$ implies feasibility of these constraints on the infinite horizon (Muske and Rawlings 1993).

5.3 Dependence on N and H_u

The number of the optimization decision variables ($Nn_x + H_u n_u$) increases with prediction and control horizon. The discrete state-space model equation and the output constraints increase with N and, with H_u in the case of input constraints. The choice

between null and range space active-set methods depends on the degrees of freedom and number of active constraints at each iteration. Therefore, if there are large number of equality, the null-space active set method is preferable for tradeoff in storage because the optimization steps are solved in the space containing the degrees of freedom vector(in most cases the manipulated variables). The Jacobian matrix of the KKT equation becomes larger and more sparse as N or H_u increases for both the active-set and primal-dual interior point methods. Therefore, the best method of solution is expected to exploit the sparse matrix structure for efficient computation.

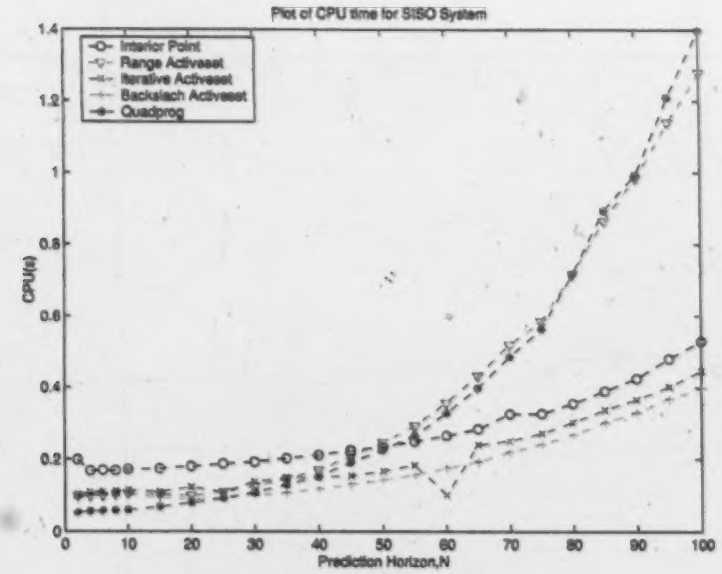
5.3.1 Off-line Computation Time

The CPU time is the amount of time the CPU is actually executing instructions. CPU times are used to compare the speed of two different processors and to measure the amount of processing time being allocated to different programs in an environment.[1] Figures 5.1 and 5.2 report the CPU times and number of iterations required by each developed optimization algorithm to solve the open-loop optimization problem associated with the case studies described in the previous chapters. In order to identify the CPU time trend of the proposed algorithms, the prediction horizon is varied and the obtained CPU times require for the execution of each algorithms in *Matlab* are recorded. Similar experiment is also carried out for the *Matlab* `quadprog`.

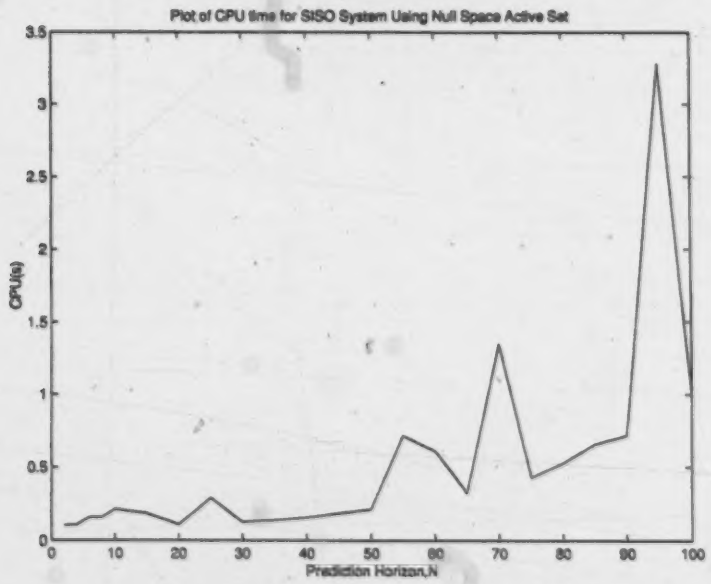
5.3.2 Flops Count

The floating point operations usually account for the computational load associated with the implementation of a numerical algorithm. Table 5.1 through 5.5 show the major operations flops per iteration breakdown for the different active-set and primal-dual IP algorithms implemented on the Shell control problem using $N = H_u = 20$. The current *Matlab* 6.1 does not execute this operation, the developed algorithms are implemented in *Matlab* 5.3 using the same machine to evaluate the scaling of each algorithm with the problem size.

[1] www.webopedia.com

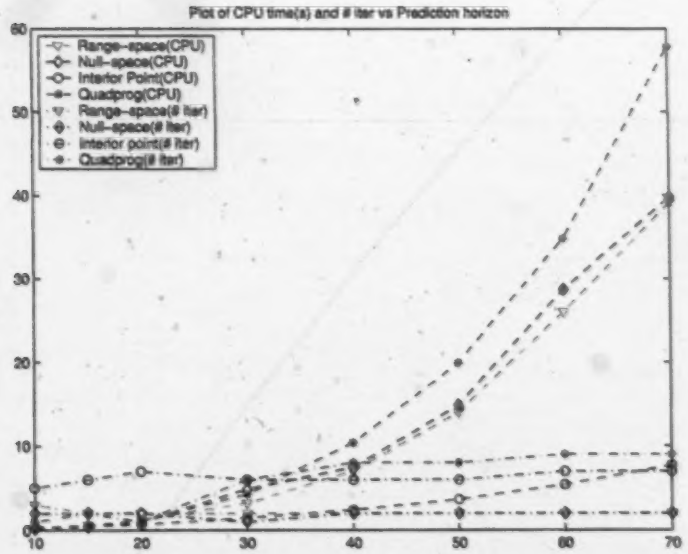


(a) CPU time trends for SISO system

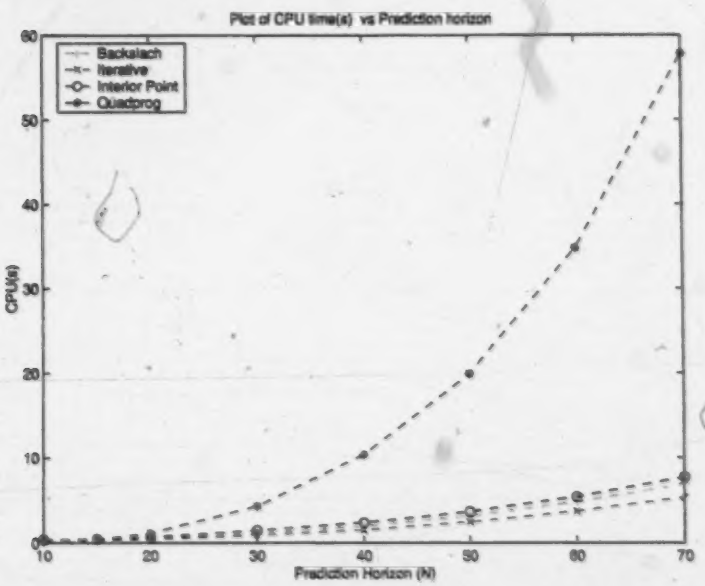


(b) CPU time trends for SISO system using Null Space Active-Set Method

Figure 5.1: CPU Trends for the Two Tank System



(a) Quadprog vs Interior Point and matrix factorization Active-set Methods



(b) Quadprog vs Interior Point and sparse matrix Active-set Methods

Figure 5.2: CPU Trends for the Shell Heavy Oil Fractionator Column

Table 5.1: Flops count using Null Space Active-set Method

Operation	Flops
QR factorization	37,204,100
Projected Hessian calculation	2,119,200
T calculation	202,282
Step direction calculation	31,200
Update	520
Total	39,557,302

Table 5.2: Flops count using Range Space Active-set Method

Operation	Flops
Hessian matrix inverse computation	49,400
QR factorization	37,204,100
$Y^T H^{-1} Y$ computation	21,624,000
Θ calculation	6,552,142
Step direction calculation	108,380
Update	520
Total	65,538,542

Table 5.3: Flops count using Backslash Operator Method

Operation	Flops
Step direction calculation	441,661
Update	520
Total	442,581

Table 5.4: Flops count using BI-CGS Method

Operation	Flops
Incomplete LU factorization	1,425,633
Residual calculation	15,380
Improvement	195,318
Step direction calculation	1,040
Step length calculation	30,361
Updates(Residuals and Solution)	1,560
Total	1,670,092

Table 5.5: Flops count using Primal-Dual IP Method

Operation	Flops
Evaluation of the stopping criteria, $\mu = \frac{\lambda^T t}{m_i}$	561
Computation of affine scaling direction	515,142
Computation of steps to the boundary, α_x and α_λ	2,247
Computation of centering-corrector steps	452,463
Computation of search directions	1,020
Step sizes determination and update	2,604
Total	974,037

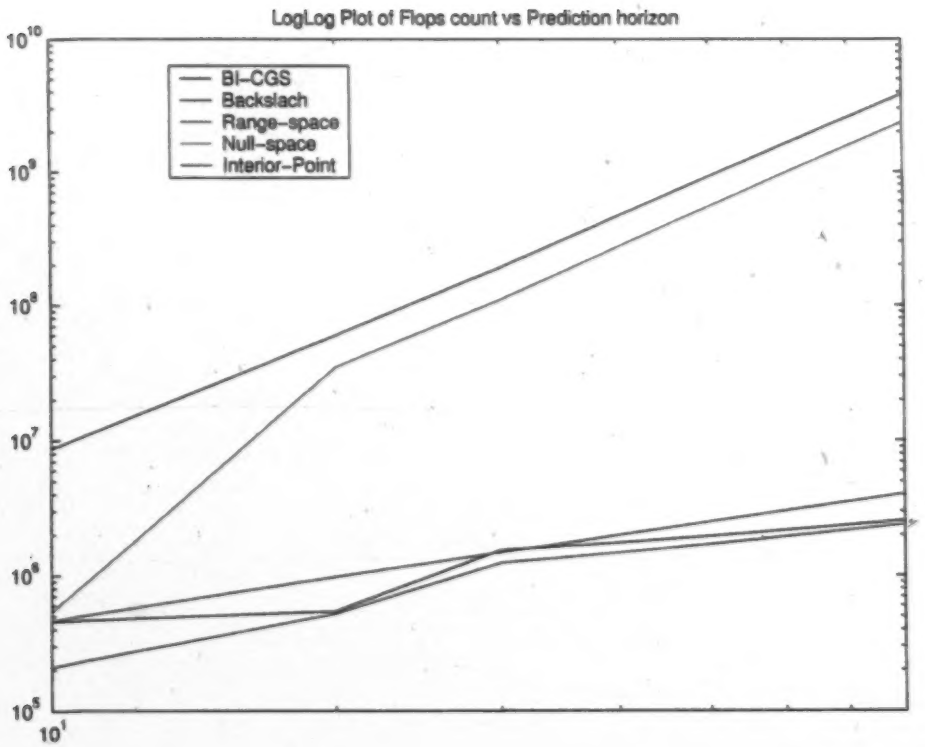


Figure 5.3: Logarithmic Plots of the Flops counts vs Prediction Horizon

Growth In Flops Count

The log-log plot in Figure 5.3 shows a linear relation between the logarithm of the flops count and the logarithm of the prediction horizon. The relationship between flops count and prediction horizon can be represented mathematically in the form below.

$$\begin{aligned} flops &= k_1 N^{k_2} \\ \ln(flops) &= \ln k_1 + k_2 \ln N \end{aligned} \tag{5.1}$$

Where k_1 is the proportionality constant and k_2 is the growth order of flops in relation with the prediction horizon N.

Range Space Method

The linearized equation representing the growth of flops count with prediction horizon for the range space active set method is obtained as

$$\ln(flops) = 9.1204 + 2.9435 \ln N \tag{5.2}$$

Thus, $flops \approx 0(N^3)$ *i.e* flops count grows in a cubic fashion with the prediction horizon.

Null Space Method

Two sets of linear equations are use to describe the growth of flops count with prediction horizon for null space method depending on the range of the prediction horizon.

$$\begin{aligned} \ln(flops) &= -0.6525 + 6.0135 \ln N,\ 1 \leq N \leq 20 \\ \ln(flops) &= 7.8745 + 3.1290 \ln N,\ 20 \leq N \leq \infty \end{aligned} \tag{5.3}$$

Thus, $flops \approx 0(N^3)$ as $N \to \infty$.

Iterative Method (Preconditioned Bi-conjugate Gradient Stabilized Method)

We have two sets of equation describing the flops count growth relative to the prediction horizon.

$$\begin{aligned} \ln(flops) &= 12.4501 + 0.2516 \ln N,\ 1 \leq N \leq 20 \\ \ln(flops) &= 12.4769 + 0.5176 \ln N,\ 20 \leq N \leq \infty \end{aligned} \tag{5.4}$$

It is shown form Eq 5.4 that as the prediction horizon increases, the flops count grows from $flops \approx 0(N^{\frac{1}{4}})$ to $flops \approx 0(N^{\frac{1}{2}})$ using the Preconditioned Bi-CGS method of solution in the active set method.

Backslash Method

This approximates a linear relation of the flops counts to the prediction horizon.

$$\ln(flops) = 9.7119 + 1.1739 \ln N \tag{5.5}$$

Thus, $flops \approx 0(N)$ using direct backslash on the sparse KKT equation.

Primal-Dual IP Method

Interior point method flops count has a linear approximation relation with the prediction horizon. The obtained equation is given below

$$\ln(flops) = 10.6668 + 1.0391 \ln N \tag{5.6}$$

Therefore $flops \approx 0(N)$ with primal-dual IP method using Krylov subspace iterative method for the linear algebra.

5.3.3 Theoretical Cost Analysis

The theoretical cost per iteration of solving QP problem using active set method reported by Bartlett *et al.* (2002) is

$$O(n(n-m)^2) + O(n(n-m)) + O((n-m)^3) \tag{5.7}$$

for null space method and

$$O(n(m^2)) + O(nm) + O(m^3) \tag{5.8}$$

for range space method.

For comparison with the experimental results, we need to relate this scaling with prediction horizon N. The scalar n represents the number of decision variable and m represents the number of linearly independent equality constraints. In this work, the MPC problem is formulated such that $n = Nn_x + H_u n_u$ and the possible number of equality constraints range as $Nn_x \leq m \leq Nn_x + Nn_y + H_u n_u$. In which n_y, n_u, and n_x represent the number of controlled output, input and state respectively. The value Nn_x is obtained from the model equation, Nn_y is obtained from the output bounds and $H_u n_u$ is obtained from the input bounds. For simplicity of this analysis we choose $m = Nn_x$ (lower bound), because this is a large horizon limiting value. The control horizon $H_u = N$ and $n = N(n_x + n_u)$. The substitution of these values for n and m in Eq 5.7 and 5.8 results to the following set of equations for the theoretical cost T_c. For null space method:

$$\begin{aligned} T_c &= O(N(n_x+n_u)(Nn_u)^2) + O(N(n_x+n_u)Nn_u) + O((Nn_u)^3) \\ &= O(N^3(n_x+n_u)n_u^2) + O(N^2(n_x+n_u)n_u) + O(N^3n_u^3) \end{aligned} \tag{5.9}$$

For range space method:

$$\begin{aligned} T_c &= O(N(n_x+n_u)(Nn_x)^2) + O(N(n_x+n_u)Nn_x) + O((Nn_x)^3) \\ &= O(N^3(n_x+n_u)n_x^2) + O(N^2(n_x+n_u)n_x) + O(N^3n_x^3) \end{aligned} \tag{5.10}$$

Therefore, the above equations show that the two active set methods scale in a cubic trend($O(N^3)$) with the prediction length theoretically.

5.4 Numerical Results

The algorithms described in Chapters 2, 3 and 4 has been implemented in *Matlab* (Version 6.1) and tested on a Pentium IV-800MHz machine running Windows 2000.

Figure 5.1(a) confirms that **quadprog** function in *Matlab* uses reduced-space active set method of solution for the MPC problem. Figure 5.1(b) shows poor scaling of the null space active set method with problem size, this is also confirmed in the null space active set method flops count growth with prediction horizon.

The results in Figure 5.2 confirms that the number of iterations for all the methods implemented grows slowly with the problem size. Figures 5.1 and 5.2 show that the CPU time required for the open loop minimization grows linearly with the prediction-horizon. In that case, there is need to keep the prediction horizon as small as possible to reduce the cost of computation.

The numerical results show that the cost of solving convex quadratic programming problem from MPC using the proposed IP algorithm increases linearly with the prediction horizon $O(N)$, this same results is obtained by Rao *et al.* (1998). The implementation of a Krylov-subspace iterative technique to active set method reduces this cost to $O(N^{\frac{1}{2}})$ for large N or even to $O(N^{\frac{1}{4}})$ when $1 \leq N \leq 20$ as shown in Eq 5.4. The computation cost is as high as $O(N^3)$ for the reduced space active set methods this is the same result obtained from the theoretical cost analysis in Section 5.3.3. As reported in Rao *et al.* (1998), the dependence of the solution time on other parameters such as the number of inputs, the number of states and the number of active constraints are cubic for both the active set and IP methods.

Tables 5.1 – 5.5 summarize the work per iteration breakdown for all the developed algorithms. These results support the CPU time plots which confirm that reduced space active set methods usually require higher time for computation when compared with either full space(sparse) active set methods or Primal-Dual IP method.

In conclusion, the implementation of the Krylov subspace iterative techniques to solve the linear algebra in the prototypical IP and active set methods reduces the computational load. The cost of computation is reduced from the prototypical $O(N^3)$ to $O(N)$ for the proposed IP method (**Interiormpc**) and $O(N^{\frac{1}{2}})$ for the proposed active set method (**Activesetconjumpc**).

5.5 Conclusions

The main conclusions on the complexity studies for efficient MPC applications are the followings:

1. **Quadprog** in *Matlab* is a poor tool for solving large-scale MPC problem because its large-scale algorithm is for problems with only bounds or equality constraints.

2. Reduced-space active set methods scale badly with problem size. The methods compute dense factors from a sparse matrix system and therefore require higher work per iteration when compared to other methods.

3. The Primal-Dual Interior Point method scales approximately linearly with problem size and takes almost the same number of iterations regardless of the problem size.

4. Reduced-space active-set methods might be preferable for dense problems, *e.g.*, for problems where the ratio of state variable to control variable is large. Any choice between either of the two reduced-space methods depends on the total number of active constraints.

5. Krylov subspace iterative method take advantage of the MPC optimization problem structure and the sparsity of the model during the computation of the optimization step. They are best suited for problems with small ratio of the state variables to control variable.

6. By exploiting the problem structure in MPC via primal-dual interior point methods and improved active set methods, through the implementation of the Krylov subspace iterative technique the cost of computation is greatly reduced, the CPU time as well as the flops count grows slowly with the prediction horizon. The computational cost is reduced from $O(N^3)$ to $O(N)$ using IP and $O(N^{\frac{1}{2}})$ using active set method.

Chapter 6

CONCLUSIONS

In the previous chapters of this thesis, different optimization algorithms for MPC applications are discussed and implemented on case studies. In this chapter, the main findings of this study are summarized and their implications for industrial and educational implementation of optimization algorithms for MPC applications are also highlighted. The first part focuses on the contribution of this thesis research and its industrial applications. Secondly, the recommendations for users and the possible future works on this subject matter are as well presented.

This work deals with the analysis of optimization algorithms for large scale Model Predictive Control(MPC). Large scale MPC problems are linearly constrained quadratic programming problems with special structures in the Hessian matrix of the objective function and the Jacobian matrix of the constraints. MPC quadratic programming problem usually results to solving a large, structured set of linear equations via efficient numerical methods. Chapter 2 deals with the development of improved active set methods to MPC. The different active set methods differ in the step direction calculation, while the reduced-space (null and range) methods focus on Jacobian matrix factorization and efficient update of factors at every iteration, the full-space (Backslash and Preconditioned Bi-CGS) methods exploit the sparse matrix structure in the optimization problem. All the active set methods solved the MPC problems accurately but differs only in the algorithm execution time. The full-space methods are more efficient than the reduced-space methods for large scale MPC applications.

Primal-Dual Interior Point method application to MPC problems is the focus in

Chapter 3. The Interior point methods are more efficient for MPC applications in which the active set changes frequently. Unlike the active set methods that proceed along the boundary of the feasible region, this method stays within the interior of the feasible region. In order to ensure strict feasibility, path-following methods are described and Mehrotra's Predictor-Corrector(MEPC) algorithm (Wright 1997*c*) is used for computation. The closed loop simulation results obtained from the implementation of the Primal-Dual IP algorithm to MPC case studies are identical to that of the active set method, thus, supporting the uniqueness of the optimization solution. The MEPC method requires more iteration steps for the open-loop minimization than the active set methods but these steps are relatively inexpensive for large sparse system of equations.

Chapter 4 deals with the *Matlab* Optimization Toolbox application to MPC problems. The quadratic programming tools in *Matlab* enforce medium-scale algorithm to large-scale MPC problem with inequality constraints or both equality and bounds constraints. The medium-scale algorithm uses the reduced space active set method for the computation.

Complexity studies is the heart of Chapter 5 with more emphasis on the scaling of the developed algorithms with MPC problem size. *Matlab* **quadprog** scales badly with the MPC problem size so also the reduced space active set methods. By exploiting the sparse matrix structure in MPC, the computational load reduces greatly as depicted by the CPU times and flops count for the sparse active set methods and the MEPC interior point method.

6.1 Contributions of Thesis

The issue of concerned is the need for efficient optimization algorithms for large scale MPC applications. The main contributions of this work include:

1. By exploiting the problem structures, Krylov subspace iterative techniques are implemented on large-scale MPC problems with both bounds and equality constraints using both the active set and primal-dual interior point methods.

2. Development of accelerated optimization softwares for speed and efficiency of

MPC application to large scale systems.

3. Reduction of the total number of iterations required for the MPC open loop minimization by implementing a least square step to obtain a good hot start for iteration.

4. Computation of the solution to wide range (SISO-MIMO systems) MPC optimization problems in a common framework.

5. Evaluation of the developed algorithms efficiencies through complexity studies.

6.2 Recommendations For Users

The following recommendations are highlighted for the users of the proposed optimization algorithms for efficient MPC applications.

1. Because of the superior ability of the primal-dual interior point approach to exploit problem structure efficiently, it should be applied to large scale multivariable MPC problems with both equalities and bound constraints.

2. Preconditioned Bi-CGS or backslash active set algorithms should be applied to large scale MPC problems with only equalities or for MPC applications in which the active constraints rarely change at every iteration because the computational cost is less and it will require small number of iterations.

3. When greater priority is placed on avoiding constraints than on obtaining the optimal decision variables, interior point algorithms should be applied, this is because IP method is strictly feasible.

4. For tradeoff in storage, null space active set algorithms should be used when number of the active constraints is close to the number of the decision variables because of the computational cost.

5. Reduced space active set algorithms should be discouraged for MPC problems with large prediction horizon, *i.e.*, when $N > 50$ from general results. This

is because at large prediction length reduced space active set methods require large number of expensive iterative steps.

The developed *Matlab* codes for all the algorithms are outlined in Appendix B and the MPC optimization decision flowchart is shown in Figure 6.1.

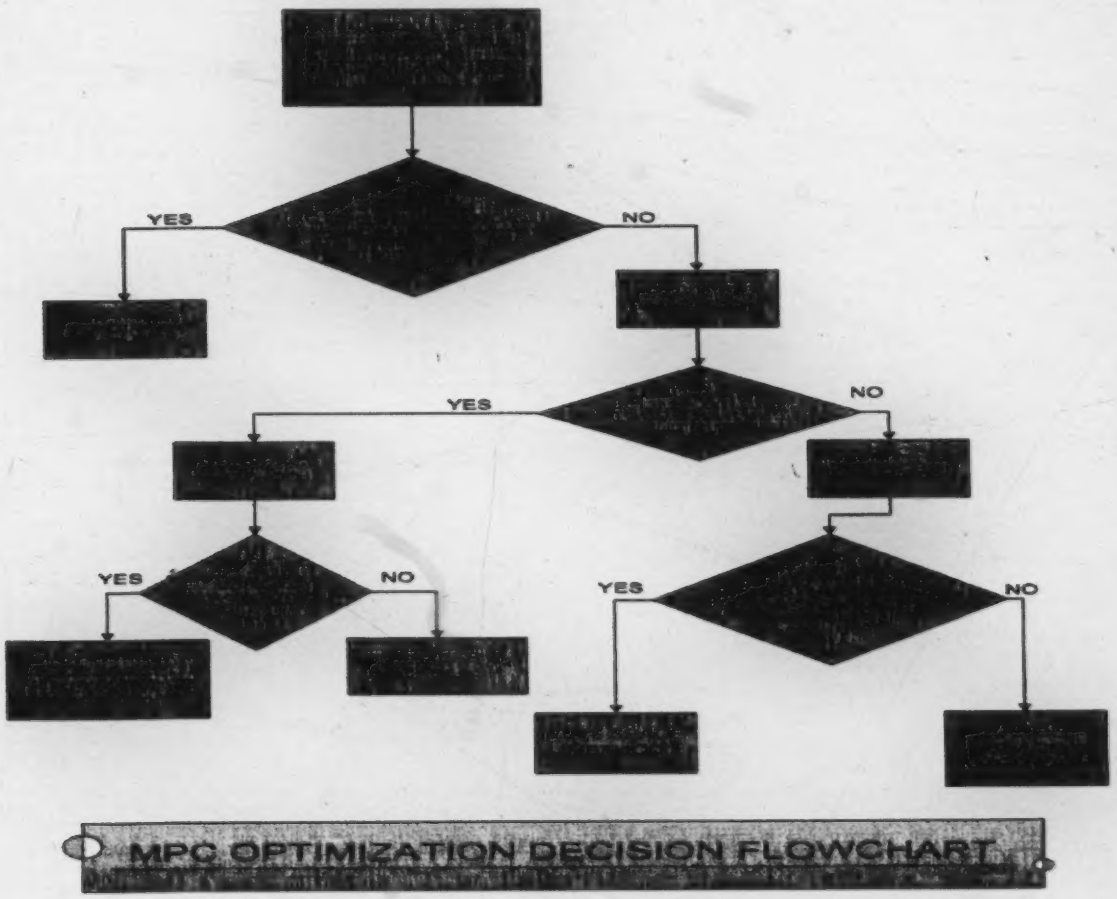


Figure 6.1: Decision Tree

6.3 Recommendations For Future Work

There has been an increasing interest recently in the area of advanced control especially in Model Predictive Control. The greatest benefits of MPC are realized in its applications to multi-variable processes with time delays, interactions and constraints. Optimization approaches for large scale Model Predictive Control is the main focus of this work. Constrained linear MPC problems are solve off-line using different optimization algorithms which are developed through improved numerical methods. Although this thesis has presented some solution strategies for MPC optimization problems, but there are some issues on MPC applications which have not been addressed. Therefore, some recommendations for future work are listed below.

1. Develop an OPC (Object Linking and Editing (OLE) for Process Control) interface to allow users easy access and usage of any of the proposed optimization softwares for MPC applications.

2. Extend this optimization methods to different type of problem class, *e.g.*, nonlinear systems.

3. Examine the influence of the plant/model mismatch on the optimal variables and simulations results.

4. Examine the sensitivity analysis of the optimization results to changes in the functional value of the process quality and safety constraints, set-points changes and changes in process model parameters.

Bibliography

Albuquerque, J., V. Gopal, G. Staus, L. T. Biegler and B. E. Ydstie (1999). Interior point SQP strategies for structured process optimization problems. *Computers and Chemical Engineering* **23**, 543–554.

Andersen, E. D. and Y. Ye (1995). On a homogeneous algorithm for the monotone complementarity problem. Technical report. Department of Management Sciences, The University of Iowa. USA.

Andersen, E. D. and Y. Ye (1998). A computational study of the homogeneous algorithm for large-scale convex optimization. *Computational Optimization and Applications* **10**(3), 243–269.

Arioli, M. (2000). The use of QR factorization in sparse quadratic programming and backward error issues. *SIAM Journal on Matrix Analysis and Applications* **21**(3), 825–839.

Arioli, M. (2001). A backward error issue analysis of a null space algorithm in sparse quadratic programming. *SIAM Journal on Matrix Analysis and Applications* **23**(2), 425–442.

Bartholomewbiggs, M. C. and M. Hernandez (1995). Using the KKT matrix in an augmented lagrangian SQP method for sparse constrained optimization. *Journal of Optimization Theory and Applications* **85**(1), 201–220.

Bartlett, Roscoe. A., Lorenz. T. Biegler and Johan Backstrom (2002). Quadratic programming algorithms for large scale model predictive control. *Journal of Process Control* **12**(7), 775–795.

Bellman, R. (1957). *Dynamic Programming.* Princeton University Press. New Jersey.

Biegler, L. T. and G. B. Sentoni (2000). Efficient formulation and solution of nonlinear model predictive control problem. *Latin American Applied Research* **30**(4), 315–324.

Blevins, T. L., G. K. McMillian, W. K. Wojsznis and M. W. Brown (2003). *Advanced Control Unleashed.* ISA. USA.

Chen, C. T. (1984). *Linear System Theory and Design.* Holt Rhinehart and Winston. New York.

Coleman, T. F. and D. C. Sorensen (1984). A note on the computation of an orthogonal basis for the null space of a matrix. *Mathematical Programming* **29**, 234–242.

Coleman, T. F. and Y. Li (1996). A reflective newton method for minimizing a quadratic function subject to bounds on some variables. *SIAM Journal of Optimization* **6**(4), 1040–1058.

D'Apuzzo, M. and M. Marino (2003). Parallel computational issues of an interior point method for solving large bound-constrained quadratic programming problems. *Parallel Computing* **29**(4), 467–483.

Dax, A. (2003). The adventures of a simple algorithm. *Linear algebra and its applications* **361**(1), 41–61.

Eiermann, M. and R. S Varga (1993). Is the optimal ω best for the SOR iteration method. *Linear Algebra and Its Applications* **182**, 257–277.

Fletcher, R. (1987). *Practical Methods of Optimization.* John Wiley and Sons. New York.

Gertz, E. M. and S. J. Wright (2003). Object-oriented software for quadratic programming. *ACM Transactions on Mathematical Software* **29**(1), 58–81.

Gill, P. E., W. Murray and M. A. Saunders (2002). SNOPT: An SQP algorithm for large scale constrained optimization. *SIAM Journal on Optimization* **12**(4), 979–1006.

Gill, P. E., Walter Murray and Margaret. H. Wright (1983). *Practical Optimization.* Academic Press. London.

Gill, P., W. Murray and M. Saunders (1995). *User's Guide for QPOPT 1.0: A Fortran Package for Quadratic Programming.* System Optimization Laboratory, Department of Operations Research, Stanford University.

Goldfarb, D. and A. Idnani (1983). A numerically stable dual method for solving convex quadratic programs. *Mathematical Programming* **27**, 1–33.

Golub, G. H. and C. F. Van Loan (1996). *Matrix Computations.* The Johns Hopkins University Press. Baltimore.

Kalman, R. E. (1960*a*). Contributions to the theory of optimal control. *BOLETIN DE LA SOCIEDAD MATEMATICA MEXICANA* **5**, 102–119.

Kalman, R. E. (1960*b*). A new approach to linear filtering and prediction problems. *ASME Journal of Basic Engineering* **82**, 34–45.

Kalman, R. E. and R.S. Bucy (1961). New results in linear filtering and prediction theory. *ASME Journal of Basic Engineering* **80**, 193–196.

Maciejowski, J. M. (2002). *Predictive Control with Constraints.* Pearson Education Limited. England.

Mathworks (1992). *Matlab High-Performance Numerical Computation and Visualization Software Reference Guide.* The Mathworks Inc. Natick.

Mathworks (2002). *Optimization Toolbox for Use with Matlab.* The Mathworks Inc. Natick.

Meadows, E. S., K. R. Muske and J. B. Rawlings (1995). Implementable model predictive control in the state space. *In Preceedings of 1995 American Control Conference* pp. 3699–3703.

Mehrotra, S. (1992*a*). Asymptotic convergence in a generalized predictor-corrector method. Technical report. Dept of Industrial Engineering and Management Science, Northwestern University. Evanston III.

Mehrotra, S. (1992*b*). On the implementations of primal-dual interior point method. *SIAM Journal of Optimization* **2**, 575–601.

Mehrotra, S. (1993). Quadratic convergence in a primal dual method. *Mathematics of Operations Research* **15**, 741.

Morari, M. and J. H. Lee (1999). Model predictive control: Past, present and future. *Computers and Chemical Engineering* **23**, 667–682.

Muske, K. R. and J. B. Rawlings (1993). Model predictive control with linear models. *AIChE* **39**(2), 262–285.

Pontryagin, L. S., V. G. Boltyanskii, R. V. Gamkrelidze and E. F. Mishchenko (1962). *The Mathematical Theory of Optimal Processes.* Wiley. New York.

Powell, M. (1983). ZQPCVX: A fortran subroutine for convex quadratic programmming. Technical report. Department of Applied Mathematics and Theoretical Physics.

Press, William H., Brian P. Flannery Saul A. Teukolsky and William T. Vetterling (1977). *Numerical Recipes in C : The Art of Scientific Computing.* Prentice-Hall. Englewood Cliffs, NJ.

Prett, D. M. and C. E. García (1988). *Fundamental Process Control.* Butterworths. Boston.

Prett, D. M. and M. Morari (1987). *The Shell Process Control Workshop. Process Control Research: Industrial and Academic Perspectives.* Butter-worths. Boston.

Rao, C. V., J. B. Rawlings and S. J. Wright (1998). Application of interior-point methods to model predictive control. *Journal of Optimization Theory and Applications* **99**, 723–757.

Rao, C.V. and J. B. Rawlings (2000). Linear programming and model predictive control. *Journal of Process Control* **10**, 283–289.

Rawlings, J. B. (2000). Tutorial overview of model predictive control. *IEEE Control System Magazine* **20**, 38–52.

Rawlings, J. B. and K. R. Muske (1993). Stability of constrained receding horizon control. *IEEE Transaction in Automatic Control* **38**(10), 1512–1516.

Santos, L. O., P. Afonso, J. Castro, N. Oliveira and L. T. Biegler (2001). On-line implementation of nonlinear MPC: An experimental case study. *Control Engineering Practice* **9**, 847–857.

Schmid, C. and L. T. Biegler (1994). Quadratic programming methods for reduced Hessian SQP. *Computer and Chemical Engineering* **18**, 817.

Ternet, D. and L. T. Biegler (1999). Interior-point methods for reduced Hessian successive quadratic programming. *Computers and Chemical Engineering* **23**(7), 859–873.

Trefethen, L. N. and D. Bau (1997). *Numerical Linear Algebra.* SIAM. Philadelphia.

Vanderbei, R. J. (1998). *LOQO: An interior point code for Quadratic Programming.* Statistical and Operations Research, Princeton University.

Vanderbei, R. J. and D. F. Shanno (1999). An interior point algorithm for nonconvex nonlinear programming. *Computational Optimization and Applications* **13**, 231–252.

Wright, M. H. (1992). *Interior point methods for constrained optimization.* Cambridge University Press. Cambridge.

Wright, S. J. (1993). Interior point methods for optimal control of discrete time systems. *Journal of Optimization Theory and Applications* **77**, 161–187.

Wright, S. J. (1997*a*). Applying new optimization algorithms to model predictive control. *AIchE Symposium Series* **93**(316), 147–155.

Wright, S. J. (1997*b*). Optimization strategies for process control. In: *Chemical Process Control.* CACHE Corporation.

Wright, S. J. (1997*c*). *Primal-Dual Interior-Point Methods.* Society for Industrial and Applied Mathematics. Philadelphia.

Wright, S. J. (1998). Ill-conditioning and computational error in interior methods for nonlinear programming. *SIAM Journal on Optimization* **9**(1), 84–111.

Zhang, Y. and D. Zhang (1995). On polynomiality of the Mehrotra type predictor-corrector interior-point algorithms. *Mathematical Programming* **68**, 303.

Zheng, A. and M. Morari (1995). Stability of model predictive control with mixed constraints. *IEEE Transaction in Automatic Control* **40**(10), 1818–1823.

Zheng, A. and M. Morari (2000). Robust control of linear time-varying systems with constraints. *International Journal of Robust and Nonlinear Control* **10**, 1063–1078.

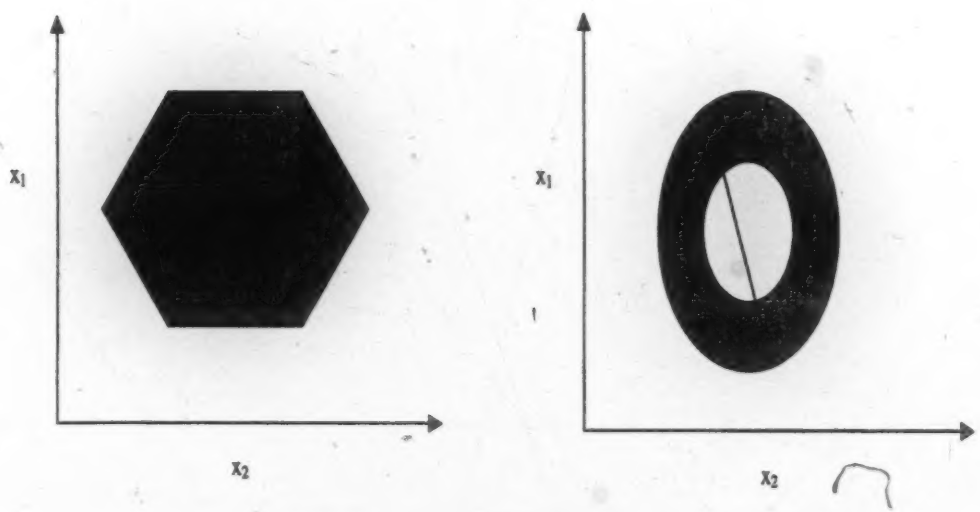
Appendix A

Mathematical Summary

A.1 DEFINITIONS

A.1.1 Convexity of a Feasible Region (Domain)

A region ∞ is convex *iff* the line joining any two points x_a and x_b in the region lies completely within the region. See Figure A.1(a)



(a) Convex Region

(b) Non-Convex Region

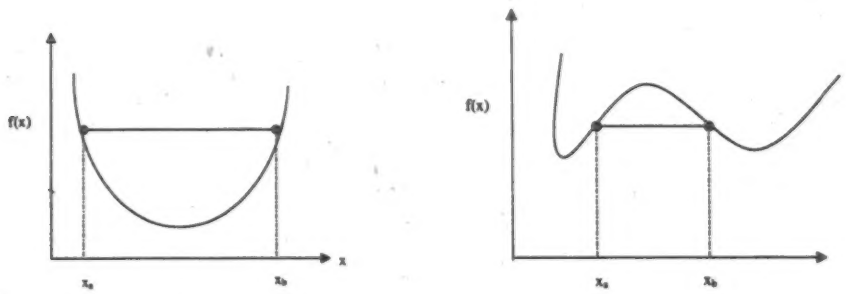
Figure A.1: Convexity of a Feasible Region

A.1.2 Convexity of a Function

A function is convex *iff* $\forall\ \beta \in [0,1]$, the line joining any two points in a convex region is greater than the function. See Figure A.2(a)

$$f(\beta x_a + (1-\beta)x_b) \ \leq\ \beta f(x_a) + (1-\beta)f(x_b),\ \forall\ x_a, x_b \in \infty \qquad \text{(A.1)}$$

The function f is strictly convex when the inequality in Eqn A.1 is strict, *i.e.*, when



(a) Convex Function (b) Non-Convex Function

Figure A.2: Convexity of a Function

"$\leq$" becomes "$<$". A quadratic function is strictly convex when its Hessian matrix is positive definite.

A.1.3 Linear Independence

A matrix is linearly dependent when a row or column contains all zeros or when a row (or column) in the matrix is linearly dependent on one or more of the other rows (or columns). By definition the columns of A, a_j, are linearly independent if

$$\sum_{j=1}^{n} d_j a_j = 0,\ \textit{iff}\ d_j = 0\ \forall j. \qquad \text{(A.2)}$$

The rank of a matrix is defined as the number of linearly independent columns ($\leq n$).

A.1.4 Degrees of Freedom

Degree of freedom analysis defines the maximum number of variables which can be independently specified to uniquely determine a feasible solution of a given problem.

Degrees of freedom =number of variables - number of linearly independent equations

A.2 SINGULAR VALUE DECOMPOSITION AND QR FACTORIZATION

A.2.1 Singular Value Decomposition

Given a real $p \times q$ matrix B with $p \geq q$, there are

- a $p \times p$ orthogonal matrix U,
- a $q \times q$ orthogonal matrix V,
- a $q \times q$ diagonal matrix Σ, with nonnegative diagonal elements arranged in decreasing order

such that

$$B = U \begin{bmatrix} \Sigma \\ 0 \end{bmatrix} V^T \tag{A.3}$$

The diagonals of Σ are the singular values, and Eq A.3 is the *singular value decomposition* (SVD) (Wright 1997*c*).

The SVD reveals a great deal about the matrix B. For instance, the rank of B is simply the number of nonzero diagonals elements in Σ. The matrix B is said to be *rank deficient* if the $rank(B) < q$.

A.2.2 Condition Number

The *condition number* of B is defined as the ratio of the largest to the smallest elements in Σ.

$$cond(B) = \Sigma_{11}/\Sigma_{qq} \tag{A.4}$$

Matrix B is singular if its *condition number* is infinite, and its ill-conditioned if its *condition number* is too large.

A.2.3 QR Factorization

Given a real $p \times q$ matrix B (with $p \geq q$), there is a $p \times p$ orthogonal matrix Q and a $q \times q$ upper triangular matrix R such that

$$B = Q \begin{bmatrix} R \\ 0 \end{bmatrix} \tag{A.5}$$

where the zero block has dimensions $(p-q) \times q$. The matrix R has zero elements on its diagonal if and only if B is rank deficient. Since Q is orthogonal, we can rewrite Eq A.5 as

$$Q^T B = \begin{bmatrix} R \\ 0 \end{bmatrix} \tag{A.6}$$

A.3 NEWTON'S METHOD FOR SYSTEMS OF NONLINEAR ALGEBRAIC EQUATION

Given a system of nonlinear equation of the form

$$f(x) = 0$$

where $f(x)$ is a vector function of x, *i.e.*,

$$\begin{aligned} f_1(x_1, x_2 \dots x_n) &= 0 \\ f_2(x_1, x_2 \dots x_n) &= 0 \\ &\dots \\ f_m(x_1, x_2 \dots x_n) &= 0 \end{aligned} \tag{A.7}$$

The equation is satisfied only at selected values of $x = r = [r_1, r_2 \dots r_n]$, called the root. Linearizing the nonlinear systems of equations using the multivariable form of the Taylor series expansion yields

$$f(x_{k+1}) \approx f(x_k) + \nabla_x f|_{x_k}(x_{k+1} - x_k) = 0 \tag{A.8}$$

where $\nabla_x f$ is called the Jacobian matrix.

$$\nabla_x f = \begin{bmatrix} \frac{\delta f_1}{\delta x_1} & \frac{\delta f_1}{\delta x_2} & \dots & \frac{\delta f_1}{\delta x_n} \\ \frac{\delta f_2}{\delta x_1} & \frac{\delta f_2}{\delta x_2} & \dots & \frac{\delta f_2}{\delta x_n} \\ \vdots & \vdots & \dots & \vdots \\ \frac{\delta f_m}{\delta x_1} & \frac{\delta f_m}{\delta x_2} & \dots & \frac{\delta f_m}{\delta x_n} \end{bmatrix} \tag{A.9}$$

Rearranging Eq A.8 gives

$$\begin{aligned} \nabla_x f|_{x_k}(x_{k+1} - x_k) &= -f(x_k) \\ \delta^{(k)} &= -\nabla_x f|_{x_k}{}^{-1} f(x_k) \end{aligned} \tag{A.10}$$

where $\delta_k = x_{k+1} - x_k$ is the displacement vector. In general, given x_k, the Newton's algorithm consists of

- Evaluating the function and the Jacobian at the current iterate x_k
- Solving the linear system for the displacement vector δ_k
- Finding the new estimate for the iterate x_{k+1} from the equations

$$\delta_k = -[\nabla_x f|_{x_k}]^{-1} f(x_k) \qquad x_{k+1} = x_k + \delta_k \qquad k = 0, 1, \dots$$

Appendix B

DEVELOPED SOFTWARE

The developed optimization software for solving large scale convex quadratic programming MPC problem are outline here for easy usage. The software are posted on the World Wide Web at

`http://www.ualberta.ca/~fadebiyi`

Please contact the author to suggest additions and modification.

B.1 Activesetnullmpc

Language: MATLAB and C

Algorithm: Null-space active set method

Input: MYMPC (Structure array consisting of the MPC problem parameters)

Linear Algebra: QR Factorization, matrix factors updates, line searches. Handles dense convex QP MPC problem. Excellent for dense MPC applications with large number of constraints.

B.2 Activesetrangempc

Language: MATLAB and C

Algorithm: Range-space active set method

Presolving: Yes

Input: MYMPC

Linear Algebra: QR Factorization, matrix factors updates, line searches. Handles

dense convex QP MPC problems. Excellent for dense MPC applications with small number of constraints.

B.3 Activesetbackmpc

Language: MATLAB and C
Algorithm: Full-space active set method
Presolving: Yes
Input: MYMPC
Linear Algebra: Sparse backslash operations and line searches. Handles large sparse convex QP MPC problems.

B.4 Activesetconjumpc

Language: MATLAB and C
Algorithm: Full-space iterative active set method
Presolving: Yes
Input: MYMPC
Linear Algebra: Preconditioned Bi-conjugate gradient stabilized method with incomplete LU Factorization and line searches. Excellent for sparse large scale MPC problem.

B.5 Interiormpc

Language: MATLAB and C
Algorithm: Modified Mehrotra's Predictor Corrector method
Presolving: Yes
Input: MYMPC
Linear Algebra: Incomplete LU Factorization, Preconditioned Bi-conjugate gradient stabilized method, line searches. Handles dense and sparse large scale convex QP MPC problem.

Appendix C

INPUT SCRIPT FILE

```
clear all
close all
clc
Ts=4;% Sampling time
steptrun=250; % Step Response Simulation Time

% Uncertainty Definition

delta=0;
delta=[-1;-1;-1;1;1];
delta=[1;-1;1;1;1];
delta=1;
delta=[-1;1;0;0;0];

% Process Model

[shellptf,dshellptf,dshellpss,shellpmod,shellpste,dmodelss,dmodelmod]
=shellfrac(Ts,steptrun,delta)
[dmodelssG,G,T,Ti]=balreal(dmodelss); dmodelss10=
modred(dmodelssG,11:size(G,1),'del');

% Steady State Definition For Reference Trajectories

M=dcgain(dmodelss10); Mc=M(:,1:3); Aeqc=Mc(1:2,:);
Beqc=zeros(2,1); Ayc=[Mc;-Mc]; Byc=0.5*ones(size(Ayc,1),1);
Z=linprog(Mc(4,:),Ayc,Byc,Aeqc,Beqc,-0.5*ones(3,1),0.5*ones(3,1));
us=Z; ys=Mc*us;

% MPC Tuning Parameters

N=20; % Prediction Horizon
Hu=20;% Control Horizon
% Penalty weight on the outputs
% Q=eye(4);
% Q=diag([20 20 1 0.001]);% for Case 1
% Q=[4 0 0 0;0 8 0 0;0 0 0.25 0;0 0  0 0.1]; %for Case 2
% Q=[5 0 0 0;0 10 0 0;0 0 0.8 0;0 0  0 0.1]; %for Case 3&4
```

```
Q=[2 0 0 0;0 4 0 0;0 0 0.2 0;0 0  0 0.001]; %for all Cases
% Penalty weight on the inputs
% R=0.1*eye(3) % for Case 1
% R=0.1*diag([0.05 2 10]);
% R=0.2*eye(3); % for Case 3
R=0.3*[2 0 0;0 1 0;0 0 1]; %for all cases
nu=3; nx=10; ny=4;
Totime=25;          % Total Closed loop Simulation Time
[A,B,C,D]=ssdata(dmodelss10);

% Initial States

in=[0 0 0]; ou=[0 0 0 0];
xnot=getxo(A,B(:,1:3),C,D(:,1:3),in',ou',1);
Bc=B(:,1:3); % For Manipulated Variables
Bd=B(:,4:5); % For Disturbances
%Step Disturbances
% d1d2=[0;0];
d1d2=[-0.5;-0.5];

% Setting up the Constraints Matrix

% Model Equation : Equality Constraints

F=[A zeros(size(A,1),nu) -eye(nx) Bc]; % factors of the constraints
M=Hu-1; for i=1:M
    j=i-1;
    k=(nx*j)+1;
    AM(k:k+nx-1,:)=[zeros(nx,(nx+nu)*(i-1)) F zeros(nx,(nx+nu)*(M-i))];
end G=[-eye(nx) Bc zeros(nx,(size(AM,2)-(nx+nu)))];
AeqHu=[G;AM]; % Equality Constraints
AeqHu=[AeqHu zeros(size(AeqHu,1),nx*(N-Hu))]; if Hu==N
    Hum=[];
    Aeq=AeqHu;
else
    Hum=[zeros(nx,(nx+nu)*(Hu-1)) A zeros(nx,nu) -eye(nx)
    zeros(nx,nx*(N-Hu-1))];
    AeqHum=[AeqHu;Hum];
    diff=N-Hu-1;
    for i=1:diff
        j=i-1;
        k=(nx*j)+1;
        AB(k:k+nx-1,:)=[zeros(nx,nx) Bc zeros(nx,nx*(i-1)) A
        -eye(nx) zeros(nx,nx*(diff-i))];
    end
    Aeq=[AeqHum;zeros(size(AB,1),(nx+nu)*(Hu-1)) AB];
end rocol=size(Aeq); count=1;
% Beq=[-A*xnot;zeros((rocol(1)-nx),1)]; % RHS vector
Beq=[(-A*xnot-Bd*d1d2);kron(ones(N-1,1),-Bd*d1d2)]; % RHS vector
P=dlyap(A,C'*Q*C); mympc =
struct('Stateweight',Q,'Inputweight',R,'Terminalweight',P,
'modelmatrixA',A,'modelmatrixBu',....
    Bc,'modelmatrixBd',Bd,'modelmatrixC',C,
    'Initialstates',xnot,'Sampletime',Totime,'Predictionhorizon',N,
    'Controlhorizon',Hu,'numinput',nu,'numoutput',ny,'numstates',
```

```
    nx,'disturbancevec',did2);

% Inequality Constraints

% Constraints on the outputs

if Hu==N
    for i=1:Hu
        j=i-1;
        k=ny*j+1;
        A1(k:k+ny-1,:)=[zeros(ny,(nx+nu)*(i-1)) C zeros(ny,
        (nx+nu)*(Hu-i)+nu)];
    end
end if Hu~=N
    for i=2:Hu
    j=i-1;
    k=ny*j+1;
    A1(k:k+ny-1,:)=[zeros(ny,(nx+nu)*(i-1)) C
    zeros(ny,(nx+nu)*(Hu-i+1)) zeros(ny,nx*(N-Hu-1))];
    end
    for i=(Hu+1):N
        j=i-1;
        k=ny*j+1;
        A1(k:k+ny-1,:)=[zeros(ny,(nx+nu)*Hu) zeros(ny,nx*(i-Hu-1))
        C zeros(ny,nx*(N-i))];
    end
end A2=-A1; AI=[A1;A2]; aI=size(AI);
BI=[-0.5*ones((aI(1)/2),1);-0.5*ones((aI(1)/2),1)];

% Creating a window for the application of the constraints on the output.

st=1; wij=st-1; stk=ny*wij+1; A1s=A1(stk:end,:); AIs=[A1s;-A1s];
aIs=size(AIs); BIs=-0.5*ones(aIs(1),1);

% Constraints on the inputs

for i=1:Hu
    j=i-1;
    k=nu*j+1;
    a1(k:k+nu-1,:)=[zeros(nu,(nx+nu)*(i-1)+nx)
    eye(nu) zeros(nu,(nx+nu)*(Hu-i)) zeros(nu,nx*(N-Hu))];
end
a2=-a1; % Upper bounds
AII=[a1;a2]; aII=size(AII);
BII=[-0.5*ones((aII(1)/2),1);-0.5*ones((aII(1)/2),1)];

% Constraints on the control moves

for i=1:Hu-1
    j=i-1;
    k=nu*j+1;
    aud1(k:k+nu-1,:)=[zeros(nu,(nx+nu)*(i-1)+nx) -eye(nu)
    zeros(nu,nx) eye(nu) zeros(nu,(nx+nu)*(Hu-1-i))
    zeros(nu,nx*(N-Hu))];
end
```

```
aud=[aud1;-aud1]; bud=-0.2*ones(size(aud,1),1);

% Hessian Matrix Calculation

H=hessiange(mympc,C); H=2*sparse(H); X=kron(ones(Hu,1),[-2*C'*Q*ys
;-2*R*us]); Y=kron(ones(N-Hu,1),-2*C'*Q*ys); d=[X;Y];
ybound=[-0.5,0.5]; ubound=[-0.5,0.5]; cbound=[-0.2,0.2];

% Save Data For Flops Calculation

save flopre
count=1;
t=cputime;

% Developed Optimization Algorithm Application

% Using Quadprog Function in \emph{Matlab}

[uopen,outopen,uf,outf,iteropen,Lamda,fvalopen,state]=
matquad(H,mympc,d,Aeq,Beq,AI,BI)

% Using the Improved Activeset Direct Backslash Method

[uopen,outopen,uf,outf,iteropen,Lamda,w,fvalopen]=
activesetshellitt(H,mympc,d,[Aeq],[Beq],
[AIs;AII],[BIs;BII],ybound,ubound)

% Using the Improved Activeset Range Space Method

[uopen,outopen,uf,outf,iteropen,Lamda,w,fvalopen]=
activesetrangeru(H,mympc,d,Aeq,Beq,
[AIs;AII],[BIs;BII],ybound,ubound)

% Using the Improved Activeset Null Space Method

[uopen,outopen,uf,outf,iteropen,Lamda,w,fvalopen,state]=
activesetnullmpcuu(H,mympc,d,[Aeq],[Beq],
[AIs;AII],[BIs;BII],ybound,ubound)

% Using the Improved Activeset BI-CGS Method

[uopen,outopen,uf,outf,iteropen,Lamda,w,fvalopen]=
activesetconjuu(H,mympc,d,Aeq,Beq,
[AIs;AII],[BIs;BII],ybound,ubound)

% Using the Improved Primal-Dual Interior Point Method

% [uopen,outopen,uf,outf,s,yi,iteropen,Lamda,fvalopen]
=interiormpcss(H,mympc,d,[Aeq],[Beq],[AI],[BI],ybound,ubound)

% Results

CPT=cputime-t % CPU time
Ts=4;
figure(1) % Manipulated and Control Variables Plots
```

```
y=[(C*xnot)';outf]; u=[in;uf]; plotall(y(:,1:3),u,Ts);
figure(2)% Outputs and Inputs Plots
plotall(y,u,Ts);
```